

BPA

Linux Reference
By: Devkumar Banerjee

Linux Filesystem Hierarchy

Version 0.65

Binh Nguyen

<linuxfilesystem@yahoo.com.au>

2004-07-30

This document outlines the set of requirements and guidelines for file and directory placement under the Linux operating system according to those of the FSSTND v2.3 final (January 29, 2004) and also its actual implementation on an arbitrary system. It is meant to be accessible to all members of the Linux community, be distribution independent and is intended to discuss the impact of the FSSTND and how it has managed to increase the efficiency of support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.

Copyright 2003 Binh Nguyen

Trademarks are owned by their owners.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Table of Contents

<u>Source and pre-formatted versions available</u>	1
<u>Chapter 1. Linux Filesystem Hierarchy</u>	2
<u>1.1. Foreword</u>	2
<u>1.2. The Root Directory</u>	6
<u>1.3. /bin</u>	7
<u>1.4. /boot</u>	9
<u>1.5. /dev</u>	10
<u>1.6. /etc</u>	15
<u>1.7. /home</u>	48
<u>1.8. /initrd</u>	49
<u>1.9. /lib</u>	50
<u>1.10. /lost+found</u>	51
<u>1.11. /media</u>	52
<u>1.12. /mnt</u>	53
<u>1.12.1. Mounting and unmounting</u>	53
<u>1.13. /opt</u>	56
<u>1.14. /proc</u>	56
<u>1.15. /root</u>	78
<u>1.16. /sbin</u>	79
<u>1.17. /usr</u>	80
<u>1.18. /var</u>	84
<u>1.19. /srv</u>	87
<u>1.20. /tmp</u>	88
<u>Glossary</u>	89
<u>Appendix A. UNIX System V Signals</u>	95
<u>Appendix B. Sources</u>	96
<u>Appendix C. About the Author</u>	99
<u>Appendix D. Contributors</u>	100
<u>Appendix E. Disclaimer</u>	101
<u>Appendix F. Donations</u>	102
<u>Appendix G. Feedback</u>	103
<u>Appendix H. GNU Free Documentation License</u>	104
<u>H.1. PREAMBLE</u>	104
<u>H.2. APPLICABILITY AND DEFINITIONS</u>	104
<u>H.3. VERBATIM COPYING</u>	105
<u>H.4. COPYING IN QUANTITY</u>	106
<u>H.5. MODIFICATIONS</u>	106
<u>H.6. COMBINING DOCUMENTS</u>	107

Table of Contents

Appendix H. GNU Free Documentation License

<u>H.7. COLLECTIONS OF DOCUMENTS</u>	108
<u>H.8. AGGREGATION WITH INDEPENDENT WORKS</u>	108
<u>H.9. TRANSLATION</u>	108
<u>H.10. TERMINATION</u>	109
<u>H.11. FUTURE REVISIONS OF THIS LICENSE</u>	109
<u>H.12. ADDENDUM: How to use this License for your documents</u>	109
<u>Notes</u>	110

Source and pre-formatted versions available

The source code and other machine readable formats of this book can be found on the Internet at the Linux Documentation Project home page <http://www.tldp.org> The latest version of this document can be found at <http://cvsview.tldp.org/index.cgi/LDP/guide/docbook/Linux-Filesystem-Hierarchy/>

Chapter 1. Linux Filesystem Hierarchy

1.1. Foreward

When migrating from another operating system such as Microsoft Windows to another; one thing that will profoundly affect the end user greatly will be the differences between the filesystems.

What are filesystems?

A *filesystem* is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk. The word is also used to refer to a partition or disk that is used to store the files or the type of the filesystem. Thus, one might say I have two filesystems meaning one has two partitions on which one stores files, or that one is using the extended filesystem, meaning the type of the filesystem.

The difference between a disk or partition and the filesystem it contains is important. A few programs (including, reasonably enough, programs that create filesystems) operate directly on the raw sectors of a disk or partition; if there is an existing file system there it will be destroyed or seriously corrupted. Most programs operate on a filesystem, and therefore won't work on a partition that doesn't contain one (or that contains one of the wrong type).

Before a partition or disk can be used as a filesystem, it needs to be initialized, and the bookkeeping data structures need to be written to the disk. This process is called *making a filesystem*.

Most UNIX filesystem types have a similar general structure, although the exact details vary quite a bit. The central concepts are *superblock*, *inode*, *data block*, *directory block*, and *indirection block*. The superblock contains information about the filesystem as a whole, such as its size (the exact information here depends on the filesystem). An inode contains all information about a file, except its name. The name is stored in the directory, together with the number of the inode. A directory entry consists of a filename and the number of the inode which represents the file. The inode contains the numbers of several data blocks, which are used to store the data in the file. There is space only for a few data block numbers in the inode, however, and if more are needed, more space for pointers to the data blocks is allocated dynamically. These dynamically allocated blocks are indirect blocks; the name indicates that in order to find the data block, one has to find its number in the indirect block first.

Like UNIX, Linux chooses to have a single hierarchical directory structure. Everything starts from the root directory, represented by /, and then expands into sub-directories instead of having so-called 'drives'. In the Windows environment, one may put one's files almost anywhere: on C drive, D drive, E drive etc. Such a file system is called a hierarchical structure and is managed by the programs themselves (program directories), not by the operating system. On the other hand, Linux sorts directories descending from the root directory / according to their importance to the boot process.

If you're wondering why Linux uses the frontslash / instead of the backslash \ as in Windows it's because it's simply following the UNIX tradition. Linux, like Unix also chooses to be case sensitive. What this means is that the case, whether in capitals or not, of the characters becomes very important. So this is not the same as THIS. This feature accounts for a fairly large proportion of problems for new users especially during file transfer operations whether it may be via removable disk media such as floppy disk or over the wire by way of FTP.

Linux Filesystem Hierarchy

The filesystem order is specific to the function of a file and not to its program context (the majority of Linux filesystems are 'Second Extended File Systems', short 'EXT2' (aka 'ext2fs' or 'extfs2') or are themselves subsets of this filesystem such as ext3 and Reiserfs). It is within this filesystem that the operating system determines into which directories programs store their files.

If you install a program in Windows, it usually stores most of its files in its own directory structure. A help file for instance may be in C:\Program Files\[program name]\ or in C:\Program Files\[program–name]\help or in C:\Program Files\[program –name]\humpty\dumpty\doo. In Linux, programs put their documentation into /usr/share/doc/[program–name], man(ual) pages into /usr/share/man/man[1–9] and info pages into /usr/share/info. They are merged into and with the system hierarchy.

As all Linux users know, unless you mount a partition or a device, the system does not know of the existence of that partition or device. This might not appear to be the easiest way to provide access to your partitions or devices, however it offers the advantage of far greater flexibility when compared to other operating systems. This kind of layout, known as the unified filesystem, does offer several advantages over the approach that Windows uses. Let's take the example of the /usr directory. This sub–directory of the root directory contains most of the system executables. With the Linux filesystem, you can choose to mount it off another partition or even off another machine over the network using an innumerable set of protocols such as NFS (Sun), Coda (CMU) or AFS (IBM). The underlying system will not and need not know the difference. The presence of the /usr directory is completely transparent. It appears to be a local directory that is part of the local directory structure.

Compliance requires that:

	shareable	unshareable
static	/usr	/etc
	/opt	/boot
variable	/var/mail	/var/run
	/var/spool/news	/var/lock

"Shareable" files are defined as those that can be stored on one host and used on others. "Unshareable" files are those that are not shareable. For example, the files in user home directories are shareable whereas device lock files are not. "Static" files include binaries, libraries, documentation files and other files that do not change without system administrator intervention. "Variable" files are defined as files that are not static.

Another reason for this unified filesystem is that Linux caches a lot of disk accesses using system memory while it is running to accelerate these processes. It is therefore vitally important that these buffers are flushed (get their content written to disk), before the system closes down. Otherwise files are left in an undetermined state which is of course a very bad thing. Flushing is achieved by 'unmounting' the partitions during proper system shutdown. In other words, don't switch your system off while it's running! You may get away with it quite often, since the Linux file system is very robust, but you may also wreak havoc upon important files. Just hit ctrl–alt–del or use the proper commands (e.g. shutdown, poweroff, init 0). This will shut down the system in a decent way which will thus, guarantee the integrity of your files.

Many of us in the Linux community have come to take for granted the existence of excellent books and documents about Linux, an example being those produced by the Linux Documentation Project. We are used to having various packages taken from different sources such as Linux FTP sites and distribution CD–ROMs

Linux Filesystem Hierarchy

integrate together smoothly. We have come to accept that we all know where critical files like mount can be found on any machine running Linux. We also take for granted CD-ROM based distributions that can be run directly from the CD and which consume only a small amount of physical hard disk or a RAM disk for some variable files like /etc/passwd, etc. This has not always been the case.

During the adolescent years of Linux during the early to mid-90s each distributor had his own favorite scheme for locating files in the directory hierarchy. Unfortunately, this caused many problems. The *Linux File System Structure* is a document, which was created to help end this anarchy. Often the group, which creates this document or the document itself, is referred to as the FSSTND. This is short for file system standard". This document has helped to standardize the layout of file systems on Linux systems everywhere. Since the original release of the standard, most distributors have adopted it in whole or in part, much to the benefit of all Linux users.

Since the first draft of the standard, the FSSTND project has been coordinated by Daniel Quinlan and development of this standard has been through consensus by a group of developers and Linux enthusiasts. The FSSTND group set out to accomplish a number of specific goals. The first goal was to solve a number of problems that existed with the current distributions at the time. Back then, it was not possible to have a shareable /usr partition, there was no clear distinction between /bin and /usr/bin, it was not possible to set up a diskless workstation, and there was just general confusion about what files went where. The second goal was to ensure the continuation of some reasonable compatibility with the de-facto standards already in use in Linux and other UNIX-like operating systems. Finally, the standard had to gain widespread approval by the developers, distributors, and users within the Linux community. Without such support, the standard would be pointless, becoming just another way of laying out the file system.

Fortunately, the FSSTND has succeeded though there are also some goals that the FSSTND project did not set out to achieve. The FSSTND does not try to emulate the scheme of any specific commercial UNIX operating system (e.g. SunOS, AIX, etc.) Furthermore, for many of the files covered by the FSSTND, the standard does not dictate whether the files should be present, merely where the files should be if they are present. Finally, for most files, the FSSTND does not attempt to dictate the format of the contents of the files. (There are some specific exceptions when several different packages may need to know the file formats to work together properly. For example, lock files that contain the process ID of the process holding the lock.) The overall objective was to establish the location where common files could be found, if they existed on a particular machine. The FSSTND project began in early August 1993. Since then, there have been a number of public revisions of this document. The latest, v2.3 was released on January 29, 2004.

If you're asking "What's the purpose of all this? Well, the answer depends on who you are. If you are a Linux user, and you don't administrate your own system then the FSSTND ensures that you will be able to find programs where you'd expect them to be if you've already had experience on another Linux machine. It also ensures that any documentation you may have makes sense. Furthermore, if you've already had some experience with Unix before, then the FSSTND shouldn't be too different from what you're currently using, with a few exceptions. Perhaps the most important thing is that the development of a standard brings Linux to a level of maturity authors and commercial application developers feel they can support.

If you administer your own machine then you gain all the benefits of the FSSTND mentioned above. You may also feel more secure in the ability of others to provide support for you, should you have a problem. Furthermore, periodic upgrades to your system are theoretically easier. Since there is an agreed-upon standard for the locations of files, package maintainers can provide instructions for upgrading that will not leave extra, older files lying around your system inhabiting valuable disk space. The FSSTND also means that there is more support from those providing source code packages for you to compile and install yourself. The provider knows, for example, where the executable for sed is to be found on a Linux machine and can use that in his installation scripts or Makefiles.

Linux Filesystem Hierarchy

If you run a large network, the FSSTND may ease many of your NFS headaches, since it specifically addresses the problems which formerly made shared implementations of /usr impractical. If you are a distributor, then you will be affected most by the Linux FSSTND. You may have to do a little extra work to make sure that your distribution is FSSTND-compliant, but your users (and hence your business) will gain by it. If your system is compliant, third party add-on packages (and possibly your own) will integrate smoothly with your system. Your users will, of course, gain all the benefits listed above, and many of your support headaches will be eased. You will benefit from all the discussion and thought that has been put into the FSSTND and avoid many of the pitfalls involved in designing a filesystem structure yourself. If you adhere to the FSSTND, you will also be able to take advantage of various features that the FSSTND was designed around. For example, the FSSTND makes "live" CD-ROMs containing everything except some of the files in the / and /var directories possible. If you write documentation for Linux, the FSSTND makes it much easier to do so, which makes sense to the Linux community. You no longer need to worry about the specific location of lock files on one distribution versus another, nor are you forced to write documentation that is only useful to the users of a specific distribution. The FSSTND is at least partly responsible for the recent explosion of Linux books being published.

If you are a developer, the existence of the FSSTND greatly eases the possibility for potential problems. You can know where important system binaries are found, so you can use them from inside your programs or your shell scripts. Supporting users is also greatly eased, since you don't have to worry about things like the location of these binaries when resolving support issues. If you are the developer of a program that needs to integrate with the rest of the system, the FSSTND ensures that you can be certain of the steps to meet this end. For example, applications such as kermit, which access the serial ports, need to know they can achieve exclusive access to the TTY device. The FSSTND specifies a common method of doing this so that all compliant applications can work together. That way you can concentrate on making more great software for Linux instead of worrying about how to detect and deal with the differences in flavors of Linux. The widespread acceptance of the FSSTND by the Linux community has been crucial to the success of both the standard and operating system. Nearly every modern distribution conforms to the Linux FSSTND. If your implementation isn't at least partially FSSTND compliant, then it is probably either very old or you built it yourself. The FSSTND itself contains a list of some of the distributions that aim to conform to the FSSTND. However, there are some distributions that are known to cut some corners in their implementation of FSSTND.

By no means does this mean that the standard itself is complete. There are still unresolved issues such as the organization of architecture-independent scripts and data files /usr/share. Up until now, the i386 has been the primary platform for Linux, so the need for standardization of such files was non-existent.

The rapid progress in porting Linux to other architectures (MC680x0, Alpha, MIPS, PowerPC) suggests that this issue will soon need to be dealt with. Another issue that is under some discussion is the creation of an /opt directory as in SVR4. The goal for such a directory would be to provide a location for large commercial or third party packages to install themselves without worrying about the requirements made by FSSTND for the other directory hierarchies. The FSSTND provides the Linux community with an excellent reference document and has proven to be an important factor in the maturation of Linux. As Linux continues to evolve, so will the FSSTND.

Now, that we have seen how things should be, let's take a look at the real world. As you will see, the implementation of this concept on Linux isn't perfect and since Linux has always attracted individualists who tend to be fairly opinionated, it has been a bone of contention among users for instance which directories certain files should be put into. With the arrival of different distributions, anarchy has once again descended upon us. Some distributions put mount directories for external media into the / directory, others into /mnt. Red Hat based distributions feature the /etc/sysconfig sub-hierarchy for configuration files concerning input and network devices. Other distributions do not have this directory at all and put the appropriate files elsewhere or

Linux Filesystem Hierarchy

even use completely different mechanisms to do the same thing. Some distributions put KDE into /opt/, others into /usr.

But even within a given file system hierarchy, there are inconsistencies. For example, even though this was never the intention of the XFree86 group, XFree86 does indeed have its own directory hierarchy.

These problems don't manifest themselves as long as you compile programs yourself. You can adapt configure scripts or Makefiles to your system's configuration or to your preference. It's a different story if you install pre-compiled packages like RPMs though. Often these are not adaptable from one file system hierarchy to another. What's worse: some RPMs might even create their own hierarchy. If you, say, install a KDE RPM from the SuSE Linux distribution on your Mandrake system, the binary will be put into /opt/kde2/bin. And thus it won't work, because Mandrake expects it to be in /usr/bin. There are of course ways to circumvent this problem but the current situation is clearly untenable. Thus, all the leading Linux distributors have joined the Linux Standard Base project, which is attempting to create a common standard for Linux distributions. This isn't easy, since changing the file system hierarchy means a lot of work for distributors so every distributor tries to push a standard which will allow them to keep as much of their own hierarchy as possible. The LSB will also encompass the proposals made by the Filesystem Hierarchy Standard project (FHS, former FSSTND).

1.2. The Root Directory

To comply with the FSSTND the following directories, or symbolic links to directories, are required in /.

/bin	Essential command binaries
/boot	Static files of the boot loader
/dev	Device files
/etc	Host-specific system configuration
/lib	Essential shared libraries and kernel modules
/media	Mount point for removable media
/mnt	Mount point for mounting a filesystem temporarily
/opt	Add-on application software packages
/sbin	Essential system binaries
/srv	Data for services provided by this system
/tmp	Temporary files
/usr	Secondary hierarchy
/var	Variable data

The following directories, or symbolic links to directories, must be in /, if the corresponding subsystem is installed:

```
/ -- the root directory
/home User home directories (optional)
/lib<qual> Alternate format essential shared libraries
          (optional)
/root Home directory for the root user (optional)
```

Each directory listed above is described in detail in separate subsections further on in this document.

The reference system will be based upon Debian 3.0r0 (Woody), 2.4.18 kernel configured to a Redhat kernel-2.4.18-i686.config file.

Hardware

Linux Filesystem Hierarchy

- ◊ Intel Celeron 766 Processor
- ◊ MSI MS-6309 V.2.0 Mainboard
- ◊ 512MB PQI PC133 SDRAM
- ◊ 16x Lite-On LTD-165H DVD-ROM
- ◊ 40x24x10 Sony CRX175A1 CD-RW
- ◊ NVIDIA RIVA 32MB TNT2 M64
- ◊ D-Link DFE-530TX 10/100 NIC
- ◊ Realtek RTL8029(AS) 10 NIC
- ◊ Lucent Mars2 Linmodem
- ◊ C-Media CMI8738 PCI Audio Device
- ◊ Miro DC-30 VIVO
- ◊ Aopen KF-45A Miditower Case
- ◊ Acer Accufeel Keyboard
- ◊ Genius Netscroll+ Mouse
- ◊ Compaq MV500 Presario Monitor

Software

- ◊ Windows XP on /dev/hda1
- ◊ FreeBSD 4.2 on /dev/hda2
- ◊ Redhat 8.0 on /dev/hda5
- ◊ Debian 3.0r0 on /dev/hda6
- ◊ Mandrake 9.1 on /dev/hda7
- ◊ Swap partition on /dev/hda8

As we all know Linux file system starts with `/`, the root directory. All other directories are 'children' of this directory. The partition which the root file system resides on is mounted first during boot and the system will not boot if it doesn't find it. On our reference system, the root directory contains the following sub-directories:

bin/ dev/ home/ lost+found/ proc/ sbin/ usr/ cdrom/ opt/ vmlinuz boot/ etc/ lib/ mnt/ root/ tmp/ var/ dvd/ floppy/ initrd/ /ftpboot

In days past it was also the home directory of 'root' but now he has been given his own directory for reasons that will be explained further on in this document.

1.3. /bin

Unlike `/sbin`, the `/bin` directory contains several useful commands that are of use to both the system administrator as well as non-privileged users. It usually contains the shells like `bash`, `csh`, etc.... and commonly used commands like `cp`, `mv`, `rm`, `cat`, `ls`. For this reason and in contrast to `/usr/bin`, the binaries in this directory are considered to be essential. The reason for this is that it contains essential system programs that must be available even if only the partition containing `/` is mounted. This situation may arise should you need to repair other partitions but have no access to shared directories (ie. you are in single user mode and hence have no network access). It also contains programs which boot scripts may depend on.

Compliance to the FSSTND means that there are no subdirectories in `/bin` and that the following commands, or symbolic links to commands, are located there.

<code>cat</code>	Utility to concatenate files to standard output
<code>chgrp</code>	Utility to change file group ownership

Linux Filesystem Hierarchy

chmod	Utility to change file access permissions
chown	Utility to change file owner and group
cp	Utility to copy files and directories
date	Utility to print or set the system date and time
dd	Utility to convert and copy a file
df	Utility to report filesystem disk space usage
dmesg	Utility to print or control the kernel message buffer
echo	Utility to display a line of text
false	Utility to do nothing, unsuccessfully
hostname	Utility to show or set the system's host name
kill	Utility to send signals to processes
ln	Utility to make links between files
login	Utility to begin a session on the system
ls	Utility to list directory contents
mkdir	Utility to make directories
mknod	Utility to make block or character special files
more	Utility to page through text
mount	Utility to mount a filesystem
mv	Utility to move/rename files
ps	Utility to report process status
pwd	Utility to print name of current working directory
rm	Utility to remove files or directories
rmdir	Utility to remove empty directories
sed	The 'sed' stream editor
sh	The Bourne command shell
stty	Utility to change and print terminal line settings
su	Utility to change user ID
sync	Utility to flush filesystem buffers
true	Utility to do nothing, successfully
umount	Utility to unmount file systems
uname	Utility to print system information

If /bin/sh is not a true Bourne shell, it must be a hard or symbolic link to the real shell command.

The rationale behind this is because sh and bash mightn't necessarily behave in the same manner. The use of a symbolic link also allows users to easily see that /bin/sh is not a true Bourne shell.

The [and test commands must be placed together in either /bin or /usr/bin.

The requirement for the [and test commands to be included as binaries (even if implemented internally by the shell) is shared with the POSIX.2 standard.

The following programs, or symbolic links to programs, must be in /bin if the corresponding subsystem is installed:

csh	The C shell (optional)
ed	The 'ed' editor (optional)
tar	The tar archiving utility (optional)
cpio	The cpio archiving utility (optional)
gzip	The GNU compression utility (optional)
gunzip	The GNU uncompression utility (optional)
zcat	The GNU uncompression utility (optional)
netstat	The network statistics utility (optional)
ping	The ICMP network test utility (optional)

If the gunzip and zcat programs exist, they must be symbolic or hard links to gzip. /bin/csh may be a symbolic link to /bin/tcsh or /usr/bin/tcsh.

The tar, gzip and cpio commands have been added to make restoration of a

Linux Filesystem Hierarchy

```
system possible (provided that / is intact).
```

Conversely, if no restoration from the root partition is ever expected, then these binaries might be omitted (e.g., a ROM chip root, mounting /usr through NFS). If restoration of a system is planned through the network, then ftp or tftp (along with everything necessary to get an ftp connection) must be available on the root partition.

1.4. /boot

This directory contains everything required for the boot process except for configuration files not needed at boot time (the most notable of those being those that belong to the GRUB boot-loader) and the map installer. Thus, the /boot directory stores data that is used before the kernel begins executing user-mode programs. This may include redundant (back-up) master boot records, sector/system map files, the kernel and other important boot files and data that is not directly edited by hand. Programs necessary to arrange for the boot loader to be able to boot a file are placed in /sbin. Configuration files for boot loaders are placed in /etc. The system kernel is located in either / or /boot (or as under Debian in /boot but is actually a symbolically linked at / in accordance with the FSSTND).

/boot/boot.0300

Backup master boot record.

/boot/boot.b

This is installed as the basic boot sector. In the case of most modern distributions it is actually a symbolic link to one of four files /boot/boot-bmp.b, /boot/boot-menu.b, /boot/boot-text.b, /boot/boot-compat.b which allow a user to change the boot-up schema so that it utilises a splash screen, a simple menu, a text based interface or a minimal boot loader to ensure compatibility respectively. In each case re-installation of lilo is necessary in order to complete the changes. To change the actual 'boot-logo' you can either use utilities such as fblogo or the more refined bootsplash.

/boot/chain.b

Used to boot non-Linux operating systems.

/boot/config-kernel-version

Installed kernel configuration. This file is most useful when compiling kernels on other systems or device modules. Below is a small sample of what the contents of the file looks like.

```
CONFIG_X86=y
CONFIG_MICROCODE=m
CONFIG_X86_MSR=m
CONFIG_MATH_EMULATION=y
CONFIG_MTRR=y
CONFIG_MODULES=y
CONFIG_MODVERSIONS=y
CONFIG_SCSI_DEBUG=m
CONFIG_I2O=m
CONFIG_ARCNET_ETH=y
CONFIG_FMV18X=m
CONFIG_HPLAN_PLUS=m
CONFIG_ETH16I=m
CONFIG_NE2000=m
CONFIG_HISAX_HFC_PCI=y
CONFIG_ISDN_DRV_AVMB1_C4=m
CONFIG_USB_RIO500=m
CONFIG_QUOTA=y
CONFIG_AUTOFS_FS=m
CONFIG_ADFS_FS=m
```

Linux Filesystem Hierarchy

```
CONFIG_AFFS_FS=m
CONFIG_HFS_FS=m
CONFIG_FAT_FS=y
CONFIG_MS DOS_FS=y
CONFIG_UMSDOS_FS=m
CONFIG_FBCON_VGA=m
CONFIG_FONT_8x8=y
CONFIG_FONT_8x16=y
CONFIG_SOUND=m
CONFIG_SOUND_CMPCI=m
CONFIG_AEDSP16=m
```

As you can see, it's rather simplistic. The line begins with the configuration option and whether it's configured as part of the kernel, as a module or not at all. Lines beginning with a # symbol are comments and are not interpreted during processing.

/boot/os2_d.b

Used to boot to the OS/2 operating system.

/boot/map

Contains the location of the kernel.

/boot/vmlinuz, /boot/vmlinuz-kernel-version

Normally the kernel or symbolic link to the kernel.

/boot/grub

This subdirectory contains the GRUB configuration files including boot-up images and sounds. GRUB is the GNU GRand Unified Bootloader, a project which intends to solve all bootup problems once and for all. One of the most interesting features, is that you don't have to install a new partition or kernel, you can change all parameters at boot time via the GRUB Console, since it knows about the filesystems.

/boot/grub/device.map

Maps devices in /dev to those used by grub. For example, (/dev/fd0) is represented by /dev/fd0 and (hd0, 4) is referenced by /dev/hda5.

/boot/grub/grub.conf, /boot/grub/menu.lst

Grub configuration file.

/boot/grub/messages

Grub boot-up welcome message.

/boot/grub/splash.xpm.gz

Grub boot-up background image.

1.5. /dev

/dev is the location of special or device files. It is a very interesting directory that highlights one important aspect of the Linux filesystem – everything is a file or a directory. Look through this directory and you should hopefully see hda1, hda2 etc.... which represent the various partitions on the first master drive of the system. /dev/cdrom and /dev/fd0 represent your CD-ROM drive and your floppy drive. This may seem strange but it will make sense if you compare the characteristics of files to that of your hardware. Both can be read from and written to. Take /dev/dsp, for instance. This file represents your speaker device. Any data written to this file will be re-directed to your speaker. If you try 'cat /boot/vmlinuz > /dev/dsp' (on a properly configured system) you should hear some sound on the speaker. That's the sound of your kernel! A file sent to /dev/lp0 gets printed. Sending data to and reading from /dev/ttys0 will allow you to communicate with a device attached there – for instance, your modem.

The majority of devices are either block or character devices; however other types of devices exist and can be created. In general, 'block devices' are devices that store or hold data, 'character devices' can be thought of as

Linux Filesystem Hierarchy

devices that transmit or transfer data. For example, diskette drives, hard drives and CD-ROM drives are all block devices while serial ports, mice and parallel printer ports are all character devices. There is a naming scheme of sorts but in the vast majority of cases these are completely illogical.

```
total 724
lrwxrwxrwx    1 root      root          13 Sep 28 18:06 MAKEDEV -> /sbin/MAKEDEV
crw-rw----   1 root      audio         14, 14 Oct  7 16:26 admmidi0
crw-rw----   1 root      audio         14, 30 Oct  7 16:26 admmidi1
lrwxrwxrwx    1 root      root          11 Oct  7 16:26 amidi -> /dev/amidi0
crw-rw----   1 root      audio         14, 13 Oct  7 16:26 amidi0
crw-rw----   1 root      audio         14, 29 Oct  7 16:26 amidi1
crw-rw----   1 root      audio         14, 11 Oct  7 16:26 amixer0
crw-rw----   1 root      audio         14, 27 Oct  7 16:26 amixer1
drwxr-xr-x    2 root      root          4096 Sep 28 18:05 ataraid
lrwxrwxrwx    1 root      root          11 Oct  7 16:26 audio -> /dev/audio0
crw-rw----   1 root      audio         14,  4 Oct  7 16:26 audio0
crw-rw----   1 root      audio         14, 20 Oct  7 16:26 audio1
crw-rw----   1 root      audio         14,  7 Mar 15 2002 audioctl
lrwxrwxrwx    1 root      root          9 Oct 14 22:51 cdrom -> /dev/scd1
lrwxrwxrwx    1 root      root          9 Oct 14 22:52 cdrom1 -> /dev/scd0
crw-----    1 root      tty           5,   1 Jan 19 20:47 console
lrwxrwxrwx    1 root      root          11 Sep 28 18:06 core -> /proc/kcore
crw-rw----   1 root      audio         14, 10 Oct  7 16:26 dmf0
crw-rw----   1 root      audio         14, 26 Oct  7 16:26 dmf1
crw-rw----   1 root      audio         14,  9 Oct  7 16:26 dmmidi0
crw-rw----   1 root      audio         14, 25 Oct  7 16:26 dmmidi1
lrwxrwxrwx    1 root      root          9 Oct  7 16:26 dsp -> /dev/dsp0
crw-rw----   1 root      audio         14,  3 Oct  7 16:26 dsp0
crw-rw----   1 root      audio         14, 19 Oct  7 16:26 dsp1
crw--w----   1 root      video        29,  0 Mar 15 2002 fb0
crw--w----   1 root      video        29,  1 Mar 15 2002 fb0autodetect
crw--w----   1 root      video        29,  0 Mar 15 2002 fb0current
crw--w----   1 root      video        29, 32 Mar 15 2002 fb1
crw--w----   1 root      video        29, 33 Mar 15 2002 fb1autodetect
crw--w----   1 root      video        29, 32 Mar 15 2002 fb1current
lrwxrwxrwx    1 root      root          13 Sep 28 18:05 fd -> /proc/self/fd
brw-rw----   1 root      floppy       2,   0 Mar 15 2002 fd0
brw-rw----   1 root      floppy       2,   1 Mar 15 2002 fd1
crw--w--w-   1 root      root          1,   7 Sep 28 18:06 full
brw-rw----   1 root      disk          3,   0 Mar 15 2002 hda
brw-rw----   1 root      disk          3,  64 Mar 15 2002 hdb
brw-rw----   1 root      disk          22,  0 Mar 15 2002 hdc
brw-rw----   1 root      disk          22, 64 Mar 15 2002 hdd
drwxr-xr-x    2 root      root          12288 Sep 28 18:05 ida
prw-----   1 root      root          0 Jan 19 20:46 initctl
brw-rw----   1 root      disk          1, 250 Mar 15 2002 initrd
drwxr-xr-x    2 root      root          4096 Sep 28 18:05 input
crw-rw----   1 root      dialout      45, 128 Mar 15 2002 ippp0
crw-rw----   1 root      dialout      45,  0 Mar 15 2002 isdn0
crw-rw----   1 root      dialout      45,  64 Mar 15 2002 isdnctrl0
crw-rw----   1 root      dialout      45, 255 Mar 15 2002 isdninfo
crw-----    1 root      root          10,  4 Mar 15 2002 jbm
crw-r-----  1 root      kmem          1,   2 Sep 28 18:06 kmem
brw-rw----   1 root      cdrom        24,  0 Mar 15 2002 lmscd
crw-----    1 root      root          10,  0 Mar 15 2002 logibm
brw-rw----   1 root      disk          7,   0 Sep 28 18:06 loop0
brw-rw----   1 root      disk          7,   1 Sep 28 18:06 loop1
crw-rw----   1 root      lp            6,   0 Mar 15 2002 lp0
crw-rw----   1 root      lp            6,   1 Mar 15 2002 lp1
crw-rw----   1 root      lp            6,   2 Mar 15 2002 lp2
crw-r-----  1 root      kmem          1,   1 Sep 28 18:06 mem
```

Linux Filesystem Hierarchy

lrwxrwxrwx	1 root	root		10 Oct 7 16:26	midi	-> /dev/midi0
crw-rw----	1 root	audio	14,	2 Oct 7 16:26	midi0	
crw-rw----	1 root	audio	14,	18 Oct 7 16:26	midi1	
lrwxrwxrwx	1 root	root		11 Oct 7 16:26	mixer	-> /dev/mixer0
crw-rw-rw-	1 root	root	14,	0 Nov 11 16:22	mixer0	
crw-rw----	1 root	audio	14,	16 Oct 7 16:26	mixer1	
lrwxrwxrwx	1 root	root		11 Oct 7 06:50	modem	-> /dev/ttyLT0
crw-rw----	1 root	audio	31,	0 Mar 15 2002	mpu401data	
crw-rw----	1 root	audio	31,	1 Mar 15 2002	mpu401stat	
crw-rw----	1 root	audio	14,	8 Oct 7 16:26	music	
crw-rw-rw-	1 root	root	1,	3 Sep 28 18:06	null	
crw-rw-rw-	1 root	root	195,	0 Jan 6 03:03	nvidia0	
crw-rw-rw-	1 root	root	195,	1 Jan 6 03:03	nvidia1	
crw-rw-rw-	1 root	root	195,	255 Jan 6 03:03	nvidiactl	
crw-rw----	1 root	lp	6,	0 Mar 15 2002	par0	
crw-rw----	1 root	lp	6,	1 Mar 15 2002	par1	
crw-rw----	1 root	lp	6,	2 Mar 15 2002	par2	
-rw-r--r--	1 root	root	665509	Oct 7 16:41	pcm	
crw-r----	1 root	kmem	1,	4 Sep 28 18:06	port	
crw-rw----	1 root	dip	108,	0 Sep 28 18:07	ppp	
crw-----	1 root	root	10,	1 Mar 15 2002	psaux	
crw-rw-rw-	1 root	root	1,	8 Sep 28 18:06	random	
crw-rw----	1 root	root	10,	135 Mar 15 2002	rtc	
brw-rw----	1 root	cdrom	11,	0 Mar 15 2002	scd0	
brw-rw----	1 root	cdrom	11,	1 Mar 15 2002	scd1	
brw-rw----	1 root	disk	8,	0 Mar 15 2002	sda	
brw-rw----	1 root	disk	8,	1 Mar 15 2002	sda1	
brw-rw----	1 root	disk	8,	2 Mar 15 2002	sda2	
brw-rw----	1 root	disk	8,	3 Mar 15 2002	sda3	
brw-rw----	1 root	disk	8,	4 Mar 15 2002	sda4	
brw-rw----	1 root	disk	8,	16 Mar 15 2002	sdb	
brw-rw----	1 root	disk	8,	17 Mar 15 2002	sdb1	
brw-rw----	1 root	disk	8,	18 Mar 15 2002	sdb2	
brw-rw----	1 root	disk	8,	19 Mar 15 2002	sdb3	
brw-rw----	1 root	disk	8,	20 Mar 15 2002	sdb4	
crw-rw----	1 root	audio	14,	1 Oct 7 16:26	sequencer	
lrwxrwxrwx	1 root	root		10 Oct 7 16:26	sequencer2	-> /dev/music
lrwxrwxrwx	1 root	root		4 Sep 28 18:05	stderr	-> fd/2
lrwxrwxrwx	1 root	root		4 Sep 28 18:05	stdin	-> fd/0
lrwxrwxrwx	1 root	root		4 Sep 28 18:05	stdout	-> fd/1
crw-rw-rw-	1 root	tty	5,	0 Sep 28 18:06	tty	
crw-----	1 root	root	4,	0 Sep 28 18:06	tty0	
crw-----	1 root	root	4,	1 Jan 19 14:59	tty1	
crw-rw----	1 root	dialout	62,	64 Oct 7 06:50	ttyLT0	
crw-rw----	1 root	dialout	4,	64 Mar 15 2002	ttyS0	
crw-rw----	1 root	dialout	4,	65 Mar 15 2002	ttyS1	
crw-rw----	1 root	dialout	4,	66 Mar 15 2002	ttyS2	
crw-rw----	1 root	dialout	4,	67 Mar 15 2002	ttyS3	
crw-rw----	1 root	dialout	188,	0 Mar 15 2002	ttyUSB0	
crw-rw----	1 root	dialout	188,	1 Mar 15 2002	ttyUSB1	
crw-----	1 root	root	1,	9 Jan 19 20:46	urandom	
drwxr-xr-x	2 root	root		4096 Sep 28 18:05	usb	
prw-r-----	1 root	adm		0 Jan 19 14:58	xconsole	
crw-rw-rw-	1 root	root	1,	5 Sep 28 18:06	zero	

Some common device files as well as their equivalent counterparts under Windows that you may wish to remember are:

/dev/ttyS0 (First communications port, COM1)

First serial port (mice, modems).

/dev/psaux (PS/2)

Linux Filesystem Hierarchy

PS/2 mouse connection (mice, keyboards).

/dev/lp0 (First printer port, LPT1)

First parallel port (printers, scanners, etc).

/dev/dsp (First audio device)

The name DSP comes from the term digital signal processor, a specialized processor chip optimized for digital signal analysis. Sound cards may use a dedicated DSP chip, or may implement the functions with a number of discrete devices. Other terms that may be used for this device are digitized voice and PCM.

/dev/usb (USB Devices)

This subdirectory contains most of the USB device nodes. Device name allocations are fairly simplistic so no elaboration is necessary.

/dev/sda (C:\, SCSI device)

First SCSI device (HDD, Memory Sticks, external mass storage devices such as CD-ROM drives on laptops, etc).

/dev/scd (D:\, SCSI CD-ROM device)

First SCSI CD-ROM device.

/dev/js0 (Standard gameport joystick)

First joystick device.

Devices are defined by type, such as 'block' or 'character', and 'major' and 'minor' number. The major number is used to categorize a device and the minor number is used to identify a specific device type. For example, all IDE device connected to the primary controller have a major number of 3. Master and slave devices, as well as individual partitions are further defined by the use of minor numbers. These are the two numbers precede the date in the following display:

```
# ls -l /dev/hd*
```

brw-rw----	1	root	disk	3,	0 Mar 15 2002	/dev/hda
brw-rw----	1	root	disk	3,	1 Mar 15 2002	/dev/hda1
brw-rw----	1	root	disk	3,	10 Mar 15 2002	/dev/hda10
brw-rw----	1	root	disk	3,	11 Mar 15 2002	/dev/hda11
brw-rw----	1	root	disk	3,	12 Mar 15 2002	/dev/hda12
brw-rw----	1	root	disk	3,	13 Mar 15 2002	/dev/hda13
brw-rw----	1	root	disk	3,	14 Mar 15 2002	/dev/hda14
brw-rw----	1	root	disk	3,	15 Mar 15 2002	/dev/hda15
brw-rw----	1	root	disk	3,	16 Mar 15 2002	/dev/hda16
brw-rw----	1	root	disk	3,	17 Mar 15 2002	/dev/hda17
brw-rw----	1	root	disk	3,	18 Mar 15 2002	/dev/hda18
brw-rw----	1	root	disk	3,	19 Mar 15 2002	/dev/hda19
brw-rw----	1	root	disk	3,	2 Mar 15 2002	/dev/hda2
brw-rw----	1	root	disk	3,	20 Mar 15 2002	/dev/hda20
brw-rw----	1	root	disk	3,	3 Mar 15 2002	/dev/hda3
brw-rw----	1	root	disk	3,	4 Mar 15 2002	/dev/hda4
brw-rw----	1	root	disk	3,	5 Mar 15 2002	/dev/hda5
brw-rw----	1	root	disk	3,	6 Mar 15 2002	/dev/hda6
brw-rw----	1	root	disk	3,	7 Mar 15 2002	/dev/hda7
brw-rw----	1	root	disk	3,	8 Mar 15 2002	/dev/hda8
brw-rw----	1	root	disk	3,	9 Mar 15 2002	/dev/hda9
brw-rw----	1	root	disk	3,	64 Mar 15 2002	/dev/hdb
brw-rw----	1	root	disk	3,	65 Mar 15 2002	/dev/hdb1
brw-rw----	1	root	disk	3,	74 Mar 15 2002	/dev/hdb10
brw-rw----	1	root	disk	3,	75 Mar 15 2002	/dev/hdb11
brw-rw----	1	root	disk	3,	76 Mar 15 2002	/dev/hdb12
brw-rw----	1	root	disk	3,	77 Mar 15 2002	/dev/hdb13
brw-rw----	1	root	disk	3,	78 Mar 15 2002	/dev/hdb14
brw-rw----	1	root	disk	3,	79 Mar 15 2002	/dev/hdb15

Linux Filesystem Hierarchy

brw-rw----	1	root	disk	3,	80	Mar 15	2002	/dev/hdb16
brw-rw----	1	root	disk	3,	81	Mar 15	2002	/dev/hdb17
brw-rw----	1	root	disk	3,	82	Mar 15	2002	/dev/hdb18
brw-rw----	1	root	disk	3,	83	Mar 15	2002	/dev/hdb19
brw-rw----	1	root	disk	3,	66	Mar 15	2002	/dev/hdb2
brw-rw----	1	root	disk	3,	84	Mar 15	2002	/dev/hdb20
brw-rw----	1	root	disk	3,	67	Mar 15	2002	/dev/hdb3
brw-rw----	1	root	disk	3,	68	Mar 15	2002	/dev/hdb4
brw-rw----	1	root	disk	3,	69	Mar 15	2002	/dev/hdb5
brw-rw----	1	root	disk	3,	70	Mar 15	2002	/dev/hdb6
brw-rw----	1	root	disk	3,	71	Mar 15	2002	/dev/hdb7
brw-rw----	1	root	disk	3,	72	Mar 15	2002	/dev/hdb8
brw-rw----	1	root	disk	3,	73	Mar 15	2002	/dev/hdb9
brw-rw----	1	root	disk	22,	0	Mar 15	2002	/dev/hdc
brw-rw----	1	root	disk	22,	64	Mar 15	2002	/dev/hdd

The major number for both hda and hdb devices is 3. Of course, the minor number changes for each specific partition. The definition of each major number category can be examined by looking at the contents of the /usr/src/linux/include/linux/major.h file. The devices.txt also documents major and minor numbers. It is located in the /usr/src/linux/Documentation directory. This file defines the major numbers. Almost all files devices are created by default at the install time. However, you can always create a device using the mknod command or the MAKEDEV script which is located in the /dev directory itself. Devices can be created with this utility by supplying the device to be created, the device type (block or character) and the major and minor numbers. For example, let's say you have accidentally deleted /dev/ttyS0 (COM1 under Windows), it can be recreated using the following command

```
# mknod ttyS0 c 4 64
```

For those of us who are rather lazy you can simply run the MAKEDEV script as such

```
# MAKEDEV *
```

which will create all devices known.

If is possible that /dev may also contain a MAKEDEV.local for the creation of any local device files.

In general and as required by the FSSTND, MAKEDEV will have provisions for creating any device that may be found on the system, not just those that a particular implementation installs.

For those of you who are wondering why Linux is using such a primitive system to reference devices its because we haven't been able to devise a sufficiently sophisticated mechanism which provides enough advantages over the current system in order to achieve widespread adoption.

To date (as of kernel version 2.4), the best attempt has been made by Richard Gooch of the CSIRO. It's called devfsd and has been a part of the kernel for a number of years now. It has been sanctioned by the kernel developers and Linus himself and details of its implementation can be found at /usr/src/linux/Documentation/filesystems/devfs/README. Below is an excerpt from this document.

Devfs is an alternative to "real" character and block special devices on your root filesystem. Kernel device drivers can register devices by name rather than major and minor numbers. These devices will appear in devfs automatically, with whatever default ownership and protection the driver specified. A daemon (devfsd) can be used to override these defaults. Devfs has been in the kernel since 2.3.46.

Linux Filesystem Hierarchy

NOTE that devfs is entirely optional. If you prefer the old disc-based device nodes, then simply leave CONFIG_DEVFS_FS=n (the default). In this case, nothing will change. ALSO NOTE that if you do enable devfs, the defaults are such that full compatibility is maintained with the old devices names.

There are two aspects to devfs: one is the underlying device namespace, which is a namespace just like any mounted filesystem. The other aspect is the filesystem code which provides a view of the device namespace. The reason I make a distinction is because devfs can be mounted many times, with each mount showing the same device namespace. Changes made are global to all mounted devfs filesystems. Also, because the devfs namespace exists without any devfs mounts, you can easily mount the root filesystem by referring to an entry in the devfs namespace.

The cost of devfs is a small increase in kernel code size and memory usage. About 7 pages of code (some of that in __init sections) and 72 bytes for each entry in the namespace. A modest system has only a couple of hundred device entries, so this costs a few more pages. Compare this with the suggestion to put /dev on a ramdisc.

On a typical machine, the cost is under 0.2 percent. On a modest system with 64 MBytes of RAM, the cost is under 0.1 percent. The accusations of "bloatware" levelled at devfs are not justified.

As of kernel version 2.6, devfs has been marked obsolete and has now been replaced by udev. A system very similar (at least from the end user's point of view) to devfs but which works entirely in userspace. An overview of udev can be found at

http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf

1.6. /etc

This is the nerve center of your system, it contains all system related configuration files in here or in its sub-directories. A "configuration file" is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary. For this reason, it's a good idea to backup this directory regularly. It will definitely save you a lot of re-configuration later if you re-install or lose your current installation. Normally, no binaries should be or are located here.

/etc/X11/

This directory tree contains all the configuration files for the X Window System. Users should note that many of the files located in this directory are actually symbolic links to the /usr/X11R6 directory tree. Thus, the presence of these files in these locations can not be certain.

/etc/X11/XF86Config, /etc/X11/XF86Config-4

The 'X' configuration file. Most modern distributions possess hardware autodetection systems that enable automatic creation of a valid file. Should autodetection fail a configuration file can also be created manually provided that you have sufficient knowledge about your system. It would be considered prudent not to attempt to type out a file from beginning to end. Rather, use common configuration utilities such as xf86config, XF86Setup and xf86cfg to create a workable template. Then, using suitable documentation commence optimization through intuition and/or trial and error. Options that can be configured via this file include X modules to be loaded on startup, keyboard, mouse, monitor and graphic chipset type. Often, commercial distributions will include their own X configuration utilities such as XDrake on Mandrake and also Xconfiguration on Redhat. Below is a sample X configuration file from the reference system

```
### BEGIN DEBCONF SECTION
# XF86Config-4 (XFree86 server configuration file) generated by dexconf, the
```

Linux Filesystem Hierarchy

```
# Debian X Configuration tool, using values from the debconf database.
#
# Edit this file with caution, and see the XF86Config-4 manual page.
# (Type "man XF86Config-4" at the shell prompt.)
#
# If you want your changes to this file preserved by dexconf, only
# make changes
# before the "### BEGIN DEBCONF SECTION" line above, and/or after the
# "### END DEBCONF SECTION" line below.
#
# To change things within the debconf section, run the command:
#   dpkg-reconfigure xserver-xfree86
# as root. Also see "How do I add custom sections to a dexconf-
# generated
# XF86Config or XF86Config-4 file?" in /usr/share/doc/xfree86-
# common/FAQ.gz.

Section "Files"
    FontPath      "unix/:7100"
# local font server
    # if the local font server has problems,
# we can fall back on these
    FontPath      "/usr/lib/X11/fonts/misc"
    FontPath      "/usr/lib/X11/fonts/cyrillic"
    FontPath      "/usr/lib/X11/fonts/100dpi/:unscaled"
    FontPath      "/usr/lib/X11/fonts/75dpi/:unscaled"
    FontPath      "/usr/lib/X11/fonts/Type1"
    FontPath      "/usr/lib/X11/fonts/Speedo"
    FontPath      "/usr/lib/X11/fonts/100dpi"
    FontPath      "/usr/lib/X11/fonts/75dpi"
EndSection

Section "Module"
    Load          "GLcore"
    Load          "bitmap"
    Load          "dbe"
    Load          "ddc"
    Load          "dri"
    Load          "extmod"
    Load          "freetype"
    Load          "glx"
    Load          "int10"
    Load          "pex5"
    Load          "record"
    Load          "speedo"
    Load          "type1"
    Load          "vbe"
    Load          "xie"
EndSection

Section "InputDevice"
    Identifier    "Generic Keyboard"
    Driver        "keyboard"
    Option        "CoreKeyboard"
    Option        "XkbRules"      "xfree86"
    Option        "XkbModel"     "pc104"
    Option        "XkbLayout"    "us"
EndSection

Section "InputDevice"
    Identifier    "Configured Mouse"
    Driver        "mouse"

```

Linux Filesystem Hierarchy

```
        Option          "CorePointer"
        Option          "Device"           "/dev/psaux"
        Option          "Protocol"        "NetMousePS/2"
        Option          "Emulate3Buttons" "true"
        Option          "ZAxisMapping"   "4 5"
EndSection

Section "InputDevice"
    Identifier      "Generic Mouse"
    Driver          "mouse"
    Option          "SendCoreEvents" "true"
    Option          "Device"         "/dev/input/mice"
    Option          "Protocol"       "ImPS/2"
    Option          "Emulate3Buttons" "true"
    Option          "ZAxisMapping"   "4 5"
EndSection

Section "Device"
    Identifier      "Generic Video Card"
    Driver          "nv"
#    Option          "UseFBDev"        "true"
    Option          "UseFBDev"        "false"
EndSection

Section "Monitor"
    Identifier      "Generic Monitor"
    HorizSync       30-38
    VertRefresh     43-95
    Option          "DPMS"
EndSection

Section "Screen"
    Identifier      "Default Screen"
    Device          "Generic Video Card"
    Monitor         "Generic Monitor"
    DefaultDepth    16
    SubSection "Display"
        Depth          1
        Modes          "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          4
        Modes          "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          8
        Modes          "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          15
        Modes          "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          16
        Modes          "800x600" "640x480"
    EndSubSection
    SubSection "Display"
        Depth          24
        Modes          "800x600" "640x480"
    EndSubSection
EndSection
```

Linux Filesystem Hierarchy

```
Section "ServerLayout"
    Identifier      "Default Layout"
    Screen          "Default Screen"
    InputDevice     "Generic Keyboard"
    InputDevice     "Configured Mouse"
    InputDevice     "Generic Mouse"
EndSection

Section "DRI"
    Mode           0666
EndSection

### END DEBCONF SECTION
```

As you can see, the layout of the file is quite simple and tends to be quite standard across most distributions. At the top are the locations of the various font files for X (note – X will not start if you do not specify a valid font), next is the "Modules" section. It details what modules are to be loaded upon startup. The most well known extensions are probably GLX (required for 3D rendering of graphics and games) and Xinerama which allows users to expand their desktop over several monitors. Next are the various "Device" sections which describe the type of hardware you have. Improper configuration of these subsections can lead to heartache and trauma with seemingly misplaced keys, bewitched mice and also constant flashing as X attempts to restart in a sometimes never ending loop. In most cases when all else fails the vesa driver seems to be able to initialise most modern video cards. In the "Screen" section it is possible to alter the default startup resolution and depth. Quite often it is possible to alter these attributes on the fly by using the alt–ctrl–+ or alt–ctrl– set of keystrokes. Lastly are the "ServerLayout" and "DRI" sections. Users will almost never touch the "DRI" section and only those who wish to utilise the Xinerama extensions of X will require having to change any of the ServerLayout options.

/etc/X11/Xmodmap

In general your default keyboard mapping comes from your X server setup. If this setup is insufficient and you are unwilling to go through the process of reconfiguration and/or you are not the superuser you'll need to use the xmodmap program. This is the utility's global configuration file.

/etc/X11/xkb/

The various symbols, types, geometries of keymaps that the X server supports can be found in this directory tree.

/etc/X11/lbxproxy/

Low Bandwidth X (LBX) proxy server configuration files. Applications that would like to take advantage of the Low Bandwidth extension to X (LBX) must make their connections to an lbxproxy. These applications need know nothing about LBX, they simply connect to the lbxproxy as if it were a regular X server. The lbxproxy accepts client connections, multiplexes them over a single connection to the X server, and performs various optimizations on the X protocol to make it faster over low bandwidth and/or high latency connections. It should be noted that such compression will not increase the pace of rendering all that much. Its primary purpose is to reduce network load and thus increase overall network latency. A competing project called DXPC (Differential X Protocol Compression) has been found to be more efficient at this task. Studies have shown though that in almost all cases ssh tunneling of X will produce far better results than through any of these specialised pieces of software.

/etc/X11/proxymngr/

X proxy services manager initialisation files. proxymngr is responsible for resolving requests from xfindproxy (in the xbase-clients package) and other similar clients, starting new proxies when appropriate, and keeping track of all the available proxy services.

/etc/X11/xdm/

X display manager configuration files. xdm manages a collection of X servers, which may be on the local host or remote machines. It provides services similar to those provided by init, getty, and login on character-based terminals: prompting for login name and password, authenticating the user, and

Linux Filesystem Hierarchy

running a session. xdm supports XDMCP (X Display Manager Control Protocol) and can also be used to run a chooser process which presents the user with a menu of possible hosts that offer XDMCP display management. If the xutils package is installed, xdm can use the sessreg utility to register login sessions to the system utmp file; this, however, is not necessary for xdm to function.

/etc/X11/xdm/xdm--config

This is the master 'xdm' configuration file. It determines where all other 'xdm' configuration files will be located. It is almost certain to be left undisturbed.

/etc/X11/gdm/

GNOME Display Manager configuration files. gdm provides the equivalent of a "login:" prompt for X displays– it pops up a login window and starts an X session. It provides all the functionality of xdm, including XDMCP support for managing remote displays. The greeting window is written using the GNOME libraries and hence looks like a GNOME application– even to the extent of supporting themes! By default, the greeter is run as an unprivileged user for security.

/etc/X11/gdm/gdm.conf

This is the primary configuration file for GDM. Through it, users can specify whether they would like their system to automatically login as a certain user, background startup image and also if they would like to run their machine as somewhat of a terminal server by using the XDMCP protocol.

/etc/X11/fonts

Home of xfs fonts.

/etc/X11/fs/

X font server configuration files. xfs is a daemon that listens on a network port and serves X fonts to X servers (and thus to X clients). All X servers have the ability to serve locally installed fonts for themselves, but xfs makes it possible to offload that job from the X server, and/or have a central repository of fonts on a networked machine running xfs so that all the machines running X servers on a network do not require their own set of fonts. xfs may also be invoked by users to, for instance, make available X fonts in user accounts that are not available to the X server or to an already running system xfs.

/etc/X11/fs/config

This is the 'xfs' initialisation file. It specifies the number of clients that are allowed to connect to the 'xfs' server at any one time, the location of log files, default resolution, the location of the fonts, etc.

```
# font server configuration file
# $Xorg: config.cpp,v 1.3 2000/08/17 19:54:19 cpqbld Exp $

# allow a maximum of 10 clients to connect to this font server
client-limit = 10
# when a font server reaches its limit, start up a new one
clone-self = on
# log messages to /var/log/xfs.log (if syslog is not used)
error-file = /var/log/xfs.log
# log errors using syslog
use-syslog = on
# turn off TCP port listening (Unix domain connections are still permitted)
no-listen = tcp
# paths to search for fonts
catalogue = /usr/lib/X11/fonts/misc/,/usr/lib/X11/fonts/cyrillic/,
/usr/lib/X11/fonts/100dpi/:unscaled,/usr/lib/X11/fonts/75dpi/:unscaled,
/usr/lib/X11/fonts/Type1/,/usr/lib/X11/fonts/CID,
/usr/lib/X11/fonts/Speedo/,/usr/lib/X11/fonts/100dpi/,
/usr/lib/X11/fonts/75dpi/
# in decipoints
default-point-size = 120
# x1,y1,x2,y2,...
default-resolutions = 100,100,75,75
```

Linux Filesystem Hierarchy

```
# font cache control, specified in kB
cache-hi-mark = 2048
cache-low-mark = 1433
cache-balance = 70
```

/etc/X11/twm

Home of configuration files for twm. The original Tabbed Window Manager.

/etc/X11/xinit/

xinit configuration files. 'xinit' is a configuration method of starting up an X session that is designed to be used as part of a script. Normally, this is used at larger sites as part of a tailored login process.

/etc/X11/xinit/xinitrc

Global xinitrc file, used by all X sessions started by xinit (startx). Its usage is of course overridden by a .xinitrc file located in the home directory of a user.

/etc/adduser.conf

'adduser' configuration. The adduser command can create new users, groups and add existing users to existing groups. Adding users with adduser is much easier than adding them by hand. Adduser will choose appropriate UID and GID values, create a home directory, copy skeletal user configuration from /etc/skel, allow you to set an initial password and the GECOS field. Optionally a custom script can be executed after this command. See adduser(8) and adduser.conf(5) for full documentation.

/etc/adjtime

Has parameters to help adjust the software (kernel) time so that it matches the RTC.

/etc/aliases

This is the aliases file – it says who gets mail for whom. It was originally generated by `eximconfig', part of the exim package distributed with Debian, but it may be edited by the mail system administrator. See exim info section for details of the things that can be configured here. An aliases database file (aliases.db) is built from the entries in the aliases files by the newaliases utility.

/etc/alternatives

It is possible for several programs fulfilling the same or similar functions to be installed on a single system at the same time. For example, many systems have several text editors installed at once. This gives choice to the users of a system, allowing each to use a different editor, if desired, but makes it difficult for a program to make a good choice of editor to invoke if the user has not specified a particular preference.

The alternatives system aims to solve this problem. A generic name in the filesystem is shared by all files providing interchangeable functionality. The alternatives system and the system administrator together determine which actual file is referenced by this generic name. For example, if the text editors ed(1) and nvi(1) are both installed on the system, the alternatives system will cause the generic name /usr/bin/editor to refer to /usr/bin/nvi by default. The system administrator can override this and cause it to refer to /usr/bin/ed instead, and the alternatives system will not alter this setting until explicitly requested to do so.

The generic name is not a direct symbolic link to the selected alternative. Instead, it is a symbolic link to a name in the alternatives directory, which in turn is a symbolic link to the actual file referenced.

This is done so that the system administrator's changes can be confined within the /etc directory.

/etc/apt

This is Debian's next generation front-end for the dpkg package manager. It provides the apt-get utility and APT dselect method that provides a simpler, safer way to install and upgrade packages. APT features complete installation ordering, multiple source capability and several other unique features, see the Users Guide in /usr/share/doc/apt/guide.text.gz

/etc/apt/sources.list

```
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-7 (20020718)]/
      unstable contrib main non-US/contrib non-US/main
```

Linux Filesystem Hierarchy

```
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-6 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-5 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-4 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-3 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-2 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
deb cdrom:[Debian GNU/Linux 3.0 r0 _Woody_ - Official i386 Binary-1 (20020718)]/  
      unstable contrib main non-US/contrib non-US/main  
  
# deb http://security.debian.org/ stable/updates main
```

Contains a list of apt-sources from which packages may be installed via APT.

/etc/asound.conf

ALSA (Advanced Linux Sound Architecture) configuration file. It is normally created via alsactl or other third-party sound configuration utilities that may be specific to a distribution such as sndconfig from Redhat.

/etc/at.deny

Users denied access to the at daemon. The 'at' command allows user to execute programs at an arbitrary time.

/etc/autoconf

Configuration files for autoconf. 'autoconf' creates scripts to configure source code packages using templates. To create configure from configure.in, run the autoconf program with no arguments. autoconf processes configure.ac with the m4 macro processor, using the Autoconf macros. If you give autoconf an argument, it reads that file instead of configure.ac and writes the configuration script to the standard output instead of to configure. If you give autoconf the argument -, it reads the standard input instead of configure.ac and writes the configuration script on the standard output.

The Autoconf macros are defined in several files. Some of the files are distributed with Autoconf; autoconf reads them first. Then it looks for the optional file acsite.m4 in the directory that contains the distributed Autoconf macro files, and for the optional file aclocal.m4 in the current directory. Those files can contain your site's or the package's own Autoconf macro definitions. If a macro is defined in more than one of the files that autoconf reads, the last definition it reads overrides the earlier ones.

/etc/bash.bashrc

System wide functions and aliases' file for interactive bash shells.

/etc/bash_completion

Programmable completion functions for bash 2.05a.

/etc/chatscripts/provider

This is the chat script used to dial out to your default service provider.

/etc/cron.d, /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly

These directories contain scripts to be executed on a regular basis by the cron daemon.

/etc/crontab

'cron' configuration file. This file is for the cron table to setup the automatic running of system routines. A cron table can also be established for individual users. The location of these user cron table files will be explained later on.

```
# /etc/crontab: system-wide crontab  
# Unlike any other crontab you don't have to run the `crontab'  
# command to install the new version when you edit this file.  
# This file also has a username field, that none of the other crontabs do.  
  
SHELL=/bin/sh
```

Linux Filesystem Hierarchy

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
25 6 * * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron || run-parts --report /etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron || run-parts --report /etc/cron.monthly
#
```

/etc/csh.login

System-wide .login file for csh(1). This file is sourced on all invocations of the shell. It contains commands that are to be executed upon login and sometimes aliases also.

/etc/csh.logout

System-wide .logout file for csh(1). This file is sourced on all invocations of the shell. It contains commands that are to be executed upon logout.

/etc/csh.cshrc

System-wide .cshrc file for csh(1). This file is sourced on all invocations of the shell. This file should contain commands to set the command search path, plus other important environment variables. This file should not contain commands that produce output or assume the shell is attached to a tty.

/etc/cups

Configuration files for the Common UNIX Printing System (CUPS). Files here are used to define client-specific parameters, such as the default server or default encryption settings.

/etc/deluser.conf

'deluser' configuration files. The deluser command can remove users and groups and remove users from a given group. Deluser can optionally remove and backup the user's home directory and mail spool or all files on the system owned by him. Optionally a custom script can also be executed after each of the commands.

/etc/devfs

This daemon sets up the /dev filesystem for use. It creates required symbolic links in /dev and also creates (if so configured, as is the default) symbolic links to the "old" names for devices.

/etc/devfs/conf.d/

'devfsd' configuration files. This daemon sets up the /dev filesystem for use. It creates required symbolic links in /dev and also creates (if so configured, as is the default) symbolic links to the "old" names for devices.

/etc/dhclient.conf, /etc/dhclient-script

'dhclient' configuration file and 'dhclient' script files respectively. It configures your system so that it may act as a client on a DHCP based network. It is essential to connect to the Internet nowadays.

/etc/dict.conf

```
#  /etc/dict.conf Written by Bob Hilliard <hilliard@debian.org>
#  1998/03/20.  Last revised Sun, 22 Nov 1998 18:10:04 -0500 This is
#  the configuration file for /usr/bin/dict.  In most cases only the
#  server keyword need be specified.

#  This default configuration will try to access a dictd server on
#  the local host, failing that, it will try the public server.  In
#  many cases this will be slow, so you should comment out the line
#  for the server that you don't want to use. To use any other
#  server, enter its IP address in place of "dict.org".

#  Refer to the dict manpage (man dict) for other options that could
#  be inserted in here.

server localhost
server dict.org
```

dict is a client for the Dictionary Server Protocol (DICT), a TCP transaction based query/response

Linux Filesystem Hierarchy

protocol that provides access to dictionary definitions from a set of natural language dictionary databases.

/etc/dosemu.conf

Configuration file for the Linux DOS Emulator. DOSEMU is a PC Emulator application that allows Linux to run a DOS operating system in a virtual x86 machine. This allows you to run many DOS applications. It includes the FreeDOS kernel, color text and full keyboard emulation (via hotkeys) via terminal, built-in X support, IBM character set font, graphics capability at the console with most compatible video cards, DPMI support so you can run DOOM, CDROM support, builtin IPX and pktdrvr support. Note – 'dosemu' is simply a ported version of Corel's own PC-DOS.

/etc/email-addresses

Part of the exim package. This file contains email addresses to use for outgoing mail. Any local part not in here will be qualified by the system domain as normal. It should contain lines of the form:

```
user: someone@isp.com
otheruser: someoneelse@anotherisp.com
```

Exim is an MTA that is considered to be rather easier to configure than smail or sendmail. It is a drop-in replacement for sendmail, mailq and rsmtip. Advanced features include the ability to reject connections from known spam sites, and an extremely efficient queue processing algorithm.

/etc/esound.conf

ESD configuration files. The Enlightened sound daemon is designed to mix together several digitized audio streams for playback by a single device. Like nasd, artsd and rplay it also has the capability to play sounds remotely.

/etc/exports

The control list of systems who want to access the system via NFS, a the list of directories that you would like to share and the permissions allocated on each share.

```
# /etc/exports: the access control list for filesystems which may be
# exported to NFS clients. See exports(5).
## LTS-begin ##

#
# The lines between the 'LTS-begin' and the 'LTS-end' were added
# on: Sun Feb 23 05:54:17 EST 2003 by the ltsp installation script.
# For more information, visit the ltsp homepage
# at http://www.ltsp.org
#

/opt/ltsp/i386           192.168.0.0/255.255.255.0(ro,no_root_squash)
/var/opt/ltsp/swapfiles   192.168.0.0/255.255.255.0(rw,no_root_squash)

#
# The following entries need to be uncommented if you want
# Local App support in ltsp
#
#/home                   192.168.0.0/255.255.255.0(rw,no_root_squash)

## LTS-end ##
```

/etc/fdprm

Floppy disk parameter table. Describes what different floppy disk formats look like. Used by setfdprm.

/etc/fstab

The configuration file for 'mount' and now 'supermount'. It lists the filesystems mounted automatically at startup by the mount -a command (in /etc/rc or equivalent startup file). Under Linux, also contains information about swap areas used automatically by swapon -a.

Linux Filesystem Hierarchy

```
# /etc/fstab: static file system information.
#
# The following is an example. Please see fstab(5) for further details.
# Please refer to mount(1) for a complete description of mount options.
#
# Format:
# <file system> <mount point> <type> <options> <dump> <pass>
#
# dump(8) uses the <dump> field to determine which file systems need
# to be dumped. fsck(8) uses the <pass> column to determine which file
# systems need to be checked--the root file system should have a 1 in
# this field, other file systems a 2, and any file systems that should
# not be checked (such as MS-DOS or NFS file systems) a 0.
#
# The `sw' option indicates that the swap partition is to be activated
# with `swapon -a'.
/dev/hda2 none swap sw 0 0
# The `bsdgroups' option indicates that the file system is to be mounted
# with BSD semantics (files inherit the group ownership of the directory
# in which they live). `ro' can be used to mount a file system read-only.
/dev/hda3 / ext2 defaults 0 1
/dev/hda5 /home ext2 defaults 0 2
/dev/hda6 /var ext2 defaults 0 2
/dev/hda7 /usr ext2 defaults,ro 0 2
/dev/hda8 /usr/local ext2 defaults,bsdgroups 0 2
# The `noauto' option indicates that the file system should not be mounted
# with `mount -a'. `user' indicates that normal users are allowed to mount
# the file system.
/dev/cdrom /cdrom iso9660 defaults,noauto,ro,user 0 0
/dev/fd0 /floppy minix defaults,noauto,user 0 0
/dev/fd1 /floppy minix defaults,noauto,user 0 0
# NFS file systems: server:
/export/usr /usr nfs defaults 0 0
# proc file system:
proc /proc proc defaults 0 0
```

/etc/ftpaccess

Determines who might get ftp-access to your machine.

/etc/ftpchroot

List of ftp users that need to be chrooted.

/etc/ftpuser

List of disallowed ftp users.

/etc/gateways

Lists gateways for 'routed'.

/etc/gettydefs

Configures console-logins.

/etc/gnome-vfs-mime-magic

MIME magic patterns as used by the Gnome VFS library.

/etc/group

Similar to /etc/passwd. It lists the configured user groups and who belongs to them.

/etc/group-

Old /etc/group file.

/etc/gshadow

Contains encrypted forms of group passwords.

/etc/gshadow-

Old /etc/gshadow file.

/etc/hostname

Linux Filesystem Hierarchy

Contains the hostname of your machine (can be fully qualified or not).

/etc/host.conf

Determines the search order for look-ups (usually hosts bind, i.e. "check /etc/hosts first and then look for a DNS").

/etc/hosts

This file is used to define a system name and domain combination with a specific IP address. This file needs to always contain an entry for an IP address, if the machine is connected to the network.

```
## etherconf DEBCONF AREA. DO NOT EDIT THIS AREA OR INSERT TEXT BEFORE IT.
127.0.0.1 localhost ::1 localhost
ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
192.168.0.99 debian.localdomain.com debian
## END OF DEBCONF AREA. PLACE YOUR EDITS BELOW; THEY WILL BE PRESERVED.
192.168.0.1 ws001
```

/etc/hosts.allow

Part of the tcp-wrappers system to control access to your machine's services. It lists hosts that are allowed to access the system and specific daemons.

```
# /etc/hosts.allow: list of hosts that are allowed to access the
# system.
# See the manual pages hosts_access(5), hosts_options(5)
# and /usr/doc/netbase/portmapper.txt.gz
#
# Example: ALL: LOCAL @some_netgroup
# ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
# If you're going to protect the portmapper use the name "portmap"
# for the daemon name. Remember that you can only use the keyword
# "ALL" and IP addresses (NOT host or domain names) for the
# portmapper. See portmap(8) and /usr/doc/portmap/portmapper.txt.gz
# for further information.
bootpd: 0.0.0.0 in.tftpd: 192.168.0.
portmap: 192.168.0.
rpc.mountd: 192.168.0.
rpc.nfsd: 192.168.0.
gdm: 192.168.0.
nasd: 192.168.0.
```

/etc/hosts.deny

part of the tcp-wrappers system to control access to your machine's services. It lists hosts that are not allowed to access the system.

```
# Example: ALL: some.host.name, .some.domain
# ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "portmap"
# for the daemon name. Remember that you can only use the keyword
# "ALL" and IP addresses (NOT host or domain names) for the
# portmapper. See portmap(8) and /usr/doc/portmap/portmapper.txt.gz
# for further information.
#
# The PARANOID wildcard matches any host whose name does not match
```

Linux Filesystem Hierarchy

```
# its address. You may wish to enable this to ensure any programs
# that don't validate looked up hostnames still leave understandable
# logs. In past versions of Debian this has been the default.
# ALL: PARANOID
```

/etc/httpd

Apache configuration files. Apache is a versatile, high-performance HTTP server. The most popular server in the world, Apache features a modular design and supports dynamic selection of extension modules at runtime. Its strong points are its range of possible customization, dynamic adjustment of the number of server processes, and a whole range of available modules including many authentication mechanisms, server-parsed HTML, server-side includes, access control, CERN httpd metafiles emulation, proxy caching, etc. Apache also supports multiple virtual homing.

/etc/identd.conf

TCP/IP IDENT protocol server. It implements the TCP/IP proposed standard IDENT user identification protocol (RFC 1413). identd operates by looking up specific TCP/IP connections and returning the username of the process owning the connection. It can also return other information besides the username.

```
# /etc/identd.conf - an example configuration file

#-- The syslog facility for error messages
# syslog:facility = daemon

#-- User and group (from passwd database) to run as
server:user = nobody

#-- Override the group id
# server:group = kmem

#-- What port to listen on when started as a daemon or from /etc/inittab
# server:port = 113

#-- The socket backlog limit
# server:backlog = 256

#-- Where to write the file containing our process id
# server:pid-file = "/var/run/identd/identd.pid"

#-- Maximum number of concurrent requests allowed (0 = unlimited)
# server:max-requests = 0

#-- Enable some protocol extensions like "VERSION" or "QUIT"
protocol:extensions = enabled

#-- Allow multiple queries per connection
protocol:multiquery = enabled

#-- Timeout in seconds since connection or last query. Zero = disable
# protocol:timeout = 120

#-- Maximum number of threads doing kernel lookups
# kernel:threads = 8

#-- Maximum number of queued kernel lookup requests
# kernel:buffers = 32

#-- Maximum number of time to retry a kernel lookup in case of failure
```

Linux Filesystem Hierarchy

```
# kernel:attempts = 5

--- Disable username lookups (only return uid numbers)
# result:uid-only = no

--- Enable the ".noident" file
# result:noident = enabled

--- Charset token to return in replies
# result:charset = "US-ASCII"

--- Opsys token to return in replies
# result:opsys = "UNIX"

--- Log all request replies to syslog (none == don't)
# result:syslog-level = none

--- Enable encryption (only available if linked with a DES library)
# result:encrypt = no

--- Path to the DES key file (only available if linked with a DES library)
# encrypt:key-file = "/usr/local/etc/identd.key"

--- Include a machine local configuration file
# include = /etc/identd.conf
```

/etc/inetd.conf

Configuration of services that are started by the INETD TCP/IP super server. 'inetd' is now deprecated. 'xinetd' has taken its place. See [/etc/xinetd.conf](#) for further details.

```
# /etc/inetd.conf: see inetd(8) for further information.
#
# Internet server configuration database
#
#
# Lines starting with "#:LABEL:" or "#<off>#" should not
# be changed unless you know what you are doing!
#
# If you want to disable an entry so it isn't touched during
# package updates just comment it out with a single '#' character.
#
# Packages should modify this file by using update-inetd(8)
#
# <service_name> <sock_type> <proto>
# <flags> <user> <server_path>
# <args>
#
#:INTERNAL: Internal services
#echo stream  tcp  nowait  root  internal
#echo dgram   udp  wait   root  internal
#chargen stream tcp  nowait  root  internal
#chargen dgram  udp   wait   root  internal
#discard stream tcp  nowait  root  internal
#discard dgram  udp   wait   root  internal
#daytime stream tcp  nowait  root  internal
#daytime dgram  udp   wait   root  internal
#time stream  tcp  nowait  root  internal
```

Linux Filesystem Hierarchy

```
#time dgram udp wait root internal

#:STANDARD: These are standard services.
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
telnet stream tcp nowait telnetd.telnetd /usr/sbin/tcpd
                                /usr/sbin/in.telnetd

#:MAIL: Mail, news and uucp services.
smtp stream tcp nowait mail /usr/sbin/exim exim -bs
imap2 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
imap3 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd

#:INFO: Info services
ident stream tcp wait identd /usr/sbin/identd identd
finger stream tcp nowait nobody /usr/sbin/tcpd
                                /usr/sbin/in.fingerd

#:BOOT: Tftp service is provided primarily for booting.
#Most sites run this only on machines acting as "boot servers."
tftp dgram udp wait nobody /usr/sbin/tcpd
                                /usr/sbin/in.tftpd -s /tftpboot
```

/etc/init.d

```
Order of scripts run in /etc/rc?.d
=====
=====
```

0. Overview.

All scripts executed by the init system are located in /etc/init.d. The directories /etc/rc?.d (? = S, 0 .. 6) contain relative links to those scripts. These links are named S<2-digit-number><original-name> or K<2-digit-number><original-name>.

If a script has the ".sh" suffix it is a bourne shell script and MAY be handled in an optimized manner. The behaviour of executing the script in an optimized way will not differ in any way from it being forked and executed in the regular way.

The following runlevels are defined:

N	System bootup (NONE).
S	Single user mode (not to be switched to directly)
0	halt
1	single user mode
2 .. 5	multi user mode
6	reboot

1. Boot.

When the system boots, the /etc/init.d/rcS script is executed. It in turn executes all the S* scripts in /etc/rcS.d in alphabetical (and thus numerical) order. The first argument passed to the executed scripts is "start". The runlevel at this point is "N" (none).

Only things that need to be run once to get the system in a consistent state are to be run. The rcS.d directory is NOT meant to replace rc.local. One should not start daemons in this runlevel unless absolutely necessary. Eg, NFS might need the portmapper, so it is OK to start it early in the boot process. But this is not the time to start the

Linux Filesystem Hierarchy

squid proxy server.

2. Going multiuser.

After the rcS.d scripts have been executed, init switches to the default runlevel as specified in /etc/inittab, usually "2".

Init then executes the /etc/init.d/rc script which takes care of starting the services in /etc/rc2.d.

Because the previous runlevel is "N" (none) the /etc/rc2.d/KXXXXXX scripts will NOT be executed - there is nothing to stop yet, the system is busy coming up.

If for example there is a service that wants to run in runlevel 4 and ONLY in that level, it will place a KXXXXXX script in /etc/rc{2,3,5}.d to stop the service when switching out of runlevel 4. We do not need to run that script at this point.

The /etc.rc2.d/SXXXXXXX scripts will be executed in alphabetical order, with the first argument set to "start".

3. Switching runlevels.

When one switches from (for example) runlevel 2 to runlevel 3, /etc/init.d/rc will first execute in alphabetical order all K scripts for runlevel 3 (/etc/rc3.d/KXXXXXX) with as first argument "stop" and then all S scripts for runlevel 3 (/etc/rc3.d/SXXXXXXX) with as first argument "start".

As an optimization, a check is made for each "service" to see if it was already running in the previous runlevel. If it was, and there is no K (stop) script present for it in the new runlevel, there is no need to start it a second time so that will not be done.

On the other hand, if there was a K script present, it is assumed the service was stopped on purpose first and so needs to be restarted.

We MIGHT make the same optimization for stop scripts as well - if no S script was present in the previous runlevel, we can assume that service was not running and we don't need to stop it either. In that case we can remove the "coming from level N" special case mentioned above in 2). But right now that has not been implemented.

4. Single user mode.

Switching to single user mode is done by switching to runlevel 1. That will cause all services to be stopped (assuming they all have a K script in /etc/rc1.d). The runlevel 1 scripts will then switch to runlevel "S" which has no scripts - all it does is spawn a shell directly on /dev/console for maintenance.

5. Halt/reboot

Going to runlevel 0 or 6 will cause the system to be halted or rebooted, respectively. For example, if we go to runlevel 6 (reboot) first all /etc/rc6.d/KXXXXXX scripts will be executed alphabetically with "stop" as the first argument.

Then the /etc/rc6.d/SXXXXXXX scripts will be executed alphabetically with "stop" as the first argument as well. The reason is that there is nothing to start any more at this point - all scripts that are

Linux Filesystem Hierarchy

run are meant to bring the system down.

In the future, the /etc/rc6.d/SXXXXXX scripts MIGHT be moved to /etc/rc6.d/K1XXXXXX for clarity.

/etc/inittab

Boot-time system configuration/initialization script. Tells init how to handle runlevels. It sets the default runlevel. This is run first except when booting in emergency (-b) mode. It also enables a user to startup a getty session on an external device such as the serial ports. To add terminals or dial-in modem lines to a system, just add more lines to /etc/inittab, one for each terminal or dial-in line. For more details, see the manual pages init, inittab, and getty. If a command fails when it starts, and init is configured to restart it, it will use a lot of system resources: init starts it, it fails, init starts it, it fails, and so on. To prevent this, init will keep track of how often it restarts a command, and if the frequency grows to high, it will delay for five minutes before restarting again. /etc/inittab also has some special features that allow init to react to special circumstances. powerwait Allows init to shut the system down, when the power fails. This assumes the use of a UPS, and software that watches the UPS and informs init that the power is off. ctrlaltdel Allows init to reboot the system, when the user presses ctrl-alt-del on the console keyboard. Note that the system administrator can configure the reaction to ctrl-alt-del to be something else instead, e.g., to be ignored, if the system is in a public location. sysinit Command to be run when the system is booted. This command usually cleans up /tmp, for example. The list above is not exhaustive. See your inittab manual page for all possibilities, and for details on how to use the ones above. To set (or reset) initial terminal colours. The following shell script should work for VGA consoles: for n in 1 2 4 5 6 7 8; do setterm -fore yellow -bold on -back blue -store > /dev/tty\$ done Substitute your favorite colors, and use /dev/ttyn for serial terminals. To make sure they are reset when people log out (if they've been changed) replace the references to getty (or mingetty or uugetty or whatever) in /etc/inittab with references to /sbin/mygetty. #!/bin/sh setterm -fore yellow -bold on -back blue -store > \$1 exec /sbin/mingetty \$@ An example /etc/inittab is provided below.

```
# /etc/inittab: init(8) configuration.
# $Id: etc.xml,v 1.10 2004/02/03 21:42:57 binh Exp $
# The default runlevel. id:2:initdefault:
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
# What to do in single-user mode.
~~:S:wait:/sbin/sulogin
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
10:0:wait:/etc/init.d/rc 0 11:1:wait:/etc/init.d/rc 1
12:2:wait:/etc/init.d/rc 2 13:3:wait:/etc/init.d/rc 3
14:4:wait:/etc/init.d/rc 4 15:5:wait:/etc/init.d/rc 5
16:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request
##--edit /etc/inittab to let this work."
# What to do when the power fails/returns.
pf::powerwait:/etc/init.d/powerfail start
```

Linux Filesystem Hierarchy

```
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
# Format:
# <id>:<runlevels>:<action>:<process>
#
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty 38400 tty1 2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3 4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5 6:23:respawn:/sbin/getty 38400 tty6
# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Undocumented features

The letters A-C can be used to spawn a daemon listed in /etc/inittab. For example, assuming you want to start getty on a port to receive a call, but only after receiving a voice call first (and not all the time). Furthermore, you want to be able to receive a data or a fax call and that when you get the voice message you'll know which you want. You insert two new lines in /etc/inittab, each with its own ID, and each with a runlevel such as A for data and B for fax. When you know which you need, you simply spawn the appropriate daemon by calling 'telinit A' or 'telinit B'.

The appropriate getty is put on the line until the first call is received. When the caller terminates the connection, the getty drops because, by definition, on demand will not respawn. The other two letters, S and Q, are special. S brings your system to maintenance mode and is the same as changing state to runlevel 1. The Q is used to tell init to reread inittab. The /etc/inittab file can be changed as often as required, but will only be read under certain circumstances: -One of its processes dies (do you need to respawn another?) -On a powerful signal from a power daemon (or a command line) -When told to change state by telinit The Q argument tells init to reread the /etc/inittab file. Even though it is called the System V runlevel system runlevels 7-9 are legitimate runlevels that can be used if necessary. The administrator must remember to alter the inittab file though and also to create the required rc?.d files.

/etc/inputrc

Global inputrc for libreadline. Readline is a function that gets a line from a user and automatically edits it.

/etc/isapnp.conf

Configuration file for ISA based cards. This standard is virtually redundant in new systems. The 'isapnptools' suite of ISA Plug-And-Play configuration utilities is used to configure such devices. These programs are suitable for all systems, whether or not they include a PnP BIOS. In fact, PnP BIOS adds some complications because it may already activate some cards so that the drivers can find them, and these tools can unconfigure them, or change their settings causing all sorts of nasty effects.

/etc/isdn

ISDN configuration files.

/etc/issue

Linux Filesystem Hierarchy

Output by getty before the login prompt. Usually contains a short description or welcoming message to the system. The contents are up to the system administrator. Debian GNU\ls 3.0 \n \l

/etc/issue.net

Presents the welcome screen to users who login remotely to your machine (whereas /etc/issue determines what a local user sees on login). Debian GNU/\%s 3.0 \%h

/etc/kde

KDE initialization scripts and KDM configuration.

/etc/kde/kdm

Location for the K Desktop Manager files. kdm manages a collection of X servers, which may be on the local host or remote machines. It provides services similar to those provided by init, getty, and login on character-based terminals: prompting for login name and password, authenticating the user, and running a session. kdm supports XDMCP (X Display Manager Control Protocol) and can also be used to run a chooser process which presents the user with a menu of possible hosts that offer XDMCP display management.

/etc/kderc

System wide KDE initialization script. Commands here executed every time the KDE environment is loaded. It's a link to /etc/kde2/system.kdeglobals

```
[Directories]
dir_config=/etc/kde2
dir_html=/usr/share/doc/kde/HTML
dir_cgi=/usr/lib/cgi-bin
dir_apps=/usr/share/applnk
dir_mime=/usr/share/mimelnk
dir_services=/usr/share/services
dir_servicetypes=/usr/share/servicetypes
[General]
TerminalApplication=x-terminal-emulator
```

/etc/ld.so.conf, /etc/ld.so.cache

/etc/ld.so.conf is a file containing a list of colon, space, tab, newline, or comma separated directories in which to search for libraries. /etc/ld.so.cache containing an ordered list of libraries found in the directories specified in /etc/ld.so.conf. This file is not in human readable format, and is not intended to be edited.

'ldconfig' creates the necessary links and cache (for use by the run-time linker, ld.so) to the most recent shared libraries found in the directories specified on the command line, in the file /etc/ld.so.conf, and in the trusted directories (/usr/lib and /lib). 'ldconfig' checks the header and file names of the libraries it encounters when determining which versions should have their links updated. ldconfig ignores symbolic links when scanning for libraries.

'ldconfig' will attempt to deduce the type of ELF libs (ie. libc5 or libc6/glibc) based on what C libs if any the library was linked against, therefore when making dynamic libraries, it is wise to explicitly link against libc (use -lc).

Some existing libs do not contain enough information to allow the deduction of their type, therefore the /etc/ld.so.conf file format allows the specification of an expected type. This is only used for those ELF libs which we can not work out. The format is like this "dirname=TYPE", where type can be libc4, libc5 or libc6. (This syntax also works on the command line). Spaces are not allowed. Also see the -p option.

Directory names containing an = are no longer legal unless they also have an expected type specifier.

'ldconfig' should normally be run by the super-user as it may require write permission on some root owned directories and files. It is normally run automatically at bootup or manually whenever new shared libraries are installed.

Linux Filesystem Hierarchy

/usr/X11R6/lib

X libraries.

/usr/local/lib

Local libraries.

/etc/lilo.conf

Configuration file for the Linux boot loader 'lilo'. 'lilo' is the original OS loader and can load Linux and others. The 'lilo' package normally contains lilo (the installer) and boot-record-images to install Linux, OS/2, DOS and generic Boot Sectors of other Oses. You can use Lilo to manage your Master Boot Record (with a simple text screen, text menu or colorful splash graphics) or call 'lilo' from other boot-loaders to jump-start the Linux kernel.

```
Prompt #Prompt user to select
OS choice at boot timeout=300  # Amount of time to wait before default OS
                                # started (in ms)
default=Debian4 #Default OS to be loaded
vga=normal #VGA mode
boot=/dev/had #location of MBR
map=/boot/map #location of kernel
install=/boot/boot-bmp.b #File to be installed as boot sector
bitmap=/boot/debian.bmp #LILO boot image
bmp-table=30p,100p,1,10 #Colours
selectable bmp-colors=13,,0,1,,0 #Colours chosen
lba32 #Required on most new systems to overcome
      #1024 cylinder problem
image=/vmlinuz #name of kernel
image label=Debian #a label
read-only #file system to be mounted read only
root=/dev/hda6 #location of root filesystem

image=/boot/bzImage
label=Debian4
read-only
root=/dev/hda6

image=/mnt/redhat/boot/vmlinuz
label=Redhat
initrd=/mnt/redhat/boot/initrd-2.4.18-14.img
read-only
root=/dev/hda5
vga=788
append=" hdc=ide-scsi hdd=ide-scsi"

image=/mnt/mandrake/boot/vmlinuz
label="Mandrake"
root=/dev/hda7
initrd=/mnt/mandrake/boot/initrd.img
append="devfs=mount hdc=ide-scsi
acpi=off quiet"
vga=788
read-only

other=/dev/hda2
table=/dev/hda
loader=/boot/chain.b
label=BSD
other=/dev/hda1
label=Windows
table=/dev/hda

other=/dev/fd0
```

Linux Filesystem Hierarchy

```
label=floppy unsafe
```

/etc/local.gen

This file lists locales that you wish to have built. You can find a list of valid supported locales at `/usr/share/i18n/SUPPORTED`. Other combinations are possible, but may not be well tested. If you change this file, you need to re-run `locale-gen`.

/etc/locale.alias

Locale name alias data base.

/etc/login.defs

Configuration control definitions for the login package. An inordinate number of attributes can be altered via this single file such as the location of mail, delay in seconds after a failed login, enabling display of fail log information, display of unknown username login failures, shell environment variables, etc....

/etc/logrotate.conf

The logrotate utility is designed to simplify the administration of log files on a system which generates a lot of log files. Logrotate allows for the automatic rotation compression, removal and mailing of log files. Logrotate can be set to handle a log file daily, weekly, monthly or when the log file gets to a certain size. Normally, logrotate runs as a daily cron job.

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

/etc/ltrace.conf

Configuration file for ltrace (Library Call Tracer). It tracks runtime library calls in dynamically linked programs. 'ltrace' is a debugging program which runs a specified command until it exits. While the command is executing, ltrace intercepts and records the dynamic library calls which are called by the executed process and the signals received by that process. It can also intercept and print the system calls executed by the program. The program to be traced need not be recompiled for this, so you can

Linux Filesystem Hierarchy

use it on binaries for which you don't have the source handy. You should install ltrace if you need a sysadmin tool for tracking the execution of processes.

/etc/magic

Magic local data and configuration file for the file(1) command. Contains the descriptions of various file formats based on which file guesses the type of the file. Insert here your local magic data. Format is described in magic(5).

/etc/mail.rc

Initialization file for 'mail'. 'mail' is an intelligent mail processing system which has a command syntax reminiscent of ed with lines replaced by messages. It's basically a command line version of Microsoft Outlook.

/etc/mailcap

'metamail' capabilities file. The mailcap file is read by the metamail program to determine how to display non-text at the local site. The syntax of a mailcap file is quite simple, at least compared to termcap files. Any line that starts with "#" is a comment. Blank lines are ignored. Otherwise, each line defines a single mailcap entry for a single content type. Long lines may be continued by ending them with a backslash character, \. Each individual mailcap entry consists of a content-type specification, a command to execute, and (possibly) a set of optional "flag" values.

/etc/mailcap.order

The mailcap ordering specifications. The order of entries in the /etc/mailcap file can be altered by editing the /etc/mailcap.order file. Each line of that file specifies a package and an optional mime type. Mailcap entries that match will be placed in the order of this file. Entries that don't match will be placed later.

/etc/mailname

Mail server hostname. Normally the same as the hostname.

/etc/menu, /etc/menu-methods

The menu package was inspired by the install-fvwm2-menu program from the old fvwm2 package. However, menu tries to provide a more general interface for menu building. With the update-menus command from this package, no package needs to be modified for every X window manager again, and it provides a unified interface for both text-and X-oriented programs.

When a package that wants to add something to the menu tree gets installed, it will run update-menus in its postinstall script. Update-menus then reads in all menu files in /etc/menu/ /usr/lib/menu and /usr/lib/menu/default, and stores the menu entries of all installed packages in memory. Once that has been done, it will run the menu-methods in /etc/menu-methods/*, and pipe the information about the menu entries to the menu-methods on stdout, so that the menu-methods can read this. Each Window Manager or other program that wants to have the debian menu tree, will supply a menu-method script in /etc/menu-methods/. This menu-method then knows how to generate the startup-file for that window manager. To facilitate this task for the window-manager maintainers, menu provides a install-menu program. This program can generate the startup files for just about every window manager.

/etc/mgetty+sendfax

Configuration files for use of mgetty as the interface on the serial port. The mgetty routine special routine has special features for handling things such as dial up connections and fax connections.

/etc/mime.types

MIME-TYPES and the extensions that represent them. This file is part of the "mime-support" package. Note: Compression schemes like "gzip", "bzip", and "compress" are not actually "mime-types". They are "encodings" and hence must not have entries in this file to map their extensions. The "mime-type" of an encoded file refers to the type of data that has been encoded, not the type of the encoding.

/etc/minicom

Linux Filesystem Hierarchy

'minicom' configuration files. 'minicom' is a communication program which somewhat resembles the shareware program TELIX but is free with source code and runs under most unices. Features include dialling directory with auto-redial, support for UUCP-style lock files on serial devices, a separate script language interpreter, capture to file, multiple users with individual configurations, and more.

/etc/modules

List of modules to be loaded at startup.

```
# /etc/modules: kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line. Comments begin with
# a "#", and everything on the line after them are ignored.
unix
af_packet
via-rhine
cmpci
ne2k-pci
nvidia
```

/etc/modules.conf

```
### This file is automatically generated by update-modules"
#
# Please do not edit this file directly. If you want to change or add
# anything please take a look at the files in /etc/modutils and read
# the manpage for update-modules.
#
### update-modules: start processing /etc/modutils/0keep
# DO NOT MODIFY THIS FILE!
# This file is not marked as conffile to make sure if you upgrade modutils
# it will be restored in case some modifications have been made.
#
# The keep command is necessary to prevent insmod and friends from ignoring
# the builtin defaults of a path-statement is encountered. Until all other
# packages use the new `add path'-statement this keep-statement is essential
# to keep your system working
keep

### update-modules: end processing /etc/modutils/0keep

### update-modules: start processing /etc/modutils/actions
# Special actions that are needed for some modules

# The BTTV module does not load the tuner module automatically,
# so do that in here
post-install bttv insmod tuner
post-remove bttv rmmod tuner

### update-modules: end processing /etc/modutils/actions

### update-modules: start processing /etc/modutils/aliases
# Aliases to tell insmod/modprobe which modules to use

# Uncomment the network protocols you don't want loaded:
# alias net-pf-1 off          # Unix
# alias net-pf-2 off          # IPv4
# alias net-pf-3 off          # Amateur Radio AX.25
# alias net-pf-4 off          # IPX
# alias net-pf-5 off          # DDP / appletalk
```

Linux Filesystem Hierarchy

```
# alias net-pf-6 off          # Amateur Radio NET/ROM
# alias net-pf-9 off          # X.25
# alias net-pf-10 off         # IPv6
# alias net-pf-11 off         # ROSE / Amateur Radio X.25 PLP
# alias net-pf-19 off         # Acorn Econet

alias char-major-10-175      agpgart
alias char-major-10-200      tun
alias char-major-81          bttv
alias char-major-108         ppp_generic
alias /dev/ppp               ppp_generic
alias tty-ldisc-3            ppp_async
alias tty-ldisc-14            ppp_synctty
alias ppp-compress-21        bsd_comp
alias ppp-compress-24        ppp_deflate
alias ppp-compress-26        ppp_deflate

# Crypto modules (see http://www.kernel.org/)
alias loop-xfer-gen-0        loop_gen
alias loop-xfer-3             loop_fish2
alias loop-xfer-gen-10        loop_gen
alias cipher-2                des
alias cipher-3                fish2
alias cipher-4                blowfish
alias cipher-6                idea
alias cipher-7                serp6f
alias cipher-8                mars6
alias cipher-11               rc62
alias cipher-15               dfc2
alias cipher-16               rijndael
alias cipher-17               rc5

### update-modules: end processing /etc/modutils/aliases

### update-modules: start processing /etc/modutils/ltmodem-2.4.18
# lt_drivers: autoloading and insertion parameter usage
alias char-major-62 lt_serial
alias /dev/tts/LT0 lt_serial
alias /dev/modem lt_serial
# options lt_modem vendor_id=0x115d device_id=0x0420 Forced=3,0x130,0x2f8
# section for lt_drivers ends

### update-modules: end processing /etc/modutils/ltmodem-2.4.18

### update-modules: start processing /etc/modutils/paths
# This file contains a list of paths that modprobe should scan,
# beside the once that are compiled into the modutils tools
# themselves.

### update-modules: end processing /etc/modutils/paths

### update-modules: start processing /etc/modutils/ppp
alias /dev/ppp               ppp_generic
alias char-major-108         ppp_generic
alias tty-ldisc-3            ppp_async
alias tty-ldisc-14            ppp_synctty
alias ppp-compress-21        bsd_comp
alias ppp-compress-24        ppp_deflate
alias ppp-compress-26        ppp_deflate
```

Linux Filesystem Hierarchy

```
### update-modules: end processing /etc/modutils/ppp

### update-modules: start processing /etc/modutils/setserial
#
# This is what I wanted to do, but logger is in /usr/bin, which isn't
# loaded when the module is first loaded into the kernel at boot time!
#
#post-install serial /etc/init.d/setserial start |
#logger -p daemon.info -t "setserial-module reload"
#pre-remove serial /etc/init.d/setserial stop |
#logger -p daemon.info -t "setserial-module unload"
#
alias /dev/tts      serial
alias /dev/tts/0    serial
alias /dev/tts/1    serial
alias /dev/tts/2    serial
alias /dev/tts/3    serial
post-install serial /etc/init.d/setserial modload > /dev/null 2> /dev/null
pre-remove serial /etc/init.d/setserial modsave > /dev/null 2> /dev/null

### update-modules: end processing /etc/modutils/setserial

### update-modules: start processing /etc/modutils/arch/i386
alias parport_lowlevel parport_pc
alias char-major-10-144 nvram
alias binfmt-0064 binfmt_aout
alias char-major-10-135 rtc

### update-modules: end processing /etc/modutils/arch/i386
```

/etc/modutils

These utilities are intended to make a Linux modular kernel manageable for all users, administrators and distribution maintainers.

/etc/mtools

Debian default mtools configuration file. The mtools series of commands work with MS-DOS files and directories on floppy disks. This allows you to use Linux with MS-DOS formatted diskettes on DOS and Windows systems.

/etc/manpath.conf

This file is used by the man_db package to configure the man and cat paths. It is also used to provide a manpath for those without one by examining their PATH environment variable. For details see the manpath(5) man page.

/etc/mediaprm

Was formally named /etc/fdprm. See /etc/fdprm for further details.

/etc/motd

The message of the day, automatically output after a successful login. Contents are up to the system administrator. Often used for getting information to every user, such as warnings about planned downtimes. Linux debian.localdomain.com 2.4.18 #1 Sat Mar 15 00:17:39 EST 2003 i686 unknown
Most of the programs included with the Debian GNU/Linux system are freely redistributable; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

/etc/mtab

List of currently mounted filesystems. Initially set up by the bootup scripts, and updated automatically by the mount command. Used when a list of mounted filesystems is needed, e.g., by the df command.

This file is sometimes a symbolic link to /proc/mounts.

/etc/networks

Linux Filesystem Hierarchy

List of networks that the system is currently located on. For example, 192.168.0.0.

/etc/nsswitch.conf

System Database/Name Service Switch configuration file.

/etc/oss.conf

OSS (Open Sound System) configuration file.

/etc/pam.d/

This directory is the home of the configuration files for PAMs, Pluggable Authentication Modules.

/etc/postfix/

Holds your postfix configuration files. Postfix is now the MTA of choice among Linux distributions.

It is sendmail-compatible, offers improved speed over sendmail, ease of administration and security.

It was originally developed by IBM and was called the IBM Secure Mailer and is used in many large commercial networks. It is now the de-facto standard.

/etc/ppp/

The place where your dial-up configuration files are placed. More than likely to be created by the text menu based pppconfig or other GUI based ppp configuration utilities such as kppp or gnome-ppp.

/etc/pam.conf

Most programs use a file under the /etc/pam.d/ directory to setup their PAM service modules. This file is and can be used, but is not recommended.

/etc/paper.config

Paper size configuration file.

/etc/papersize

Default papersize.

/etc/passwd

This is the 'old' password file, It is kept for compatibility and contains the user database, with fields giving the username, real name, home directory, encrypted password, and other information about each user. The format is documented in the passwd man(ual) page.

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:100:sync:/bin:/bin/sync games:x:5:100:games:/usr/games:/bin/sh
man:x:6:100:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh
postgres:x:31:32:postgres:/var/lib/postgres:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
operator:x:37:37:Operator:/var:/bin/sh
list:x:38:38:SmartList:/var/list:/bin/sh irc:x:39:39:ircd:/var:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/home:/bin/sh
binh:x:1000:1000:,,,:/home/binh:/bin/bash
identd:x:100:65534::/var/run/identd:/bin/false
sshd:x:101:65534::/var/run/sshd:/bin/false gdm:x:102:101:Gnome Display
Manager:/var/lib/gdm:/bin/false
telnetd:x:103:103::/usr/lib/telnetd:/bin/false
dummy:x:1001:1001:,,,:/home/dummy:/bin/bash
```

/etc/passwd--

Old /etc/passwd file.

/etc/printcap

Printer configuration (capabilities) file. The definition of all system printers, whether local or remote, is stored in this file. Its layout is similar to that of /etc/termcap but it uses a different syntax.

/etc/profile

Files and commands to be executed at login or startup time by the Bourne or C shells. These allow the system administrator to set global defaults for all users.

Linux Filesystem Hierarchy

/etc/profile.d

Shells scripts to be executed upon login to the Bourne or C shells. These scripts are normally called from the /etc/profile file.

/etc/protocols

Protocols definitions file. It describes the various DARPA Internet protocols that are available from the TCP/IP subsystem. It should be consulted instead of using the numbers in the ARPA include files or resorting to guesstimation. This file should be left untouched since changes could result in incorrect IP packages.

/etc/pcmcia

Configuration files for PCMCIA devices. Generally only useful to laptop users.

/etc/reportbug.conf

Configuration file for reportbug. Reportbug is primarily designed to report bugs in the Debian distribution. By default it creates an e-mail to the Debian bug tracking system at mit@bugs.debian.org with information about the bug. Using the –bts option you can report bugs to other servers also using ddebbugs such as KDE.org. It is similar to bug but has far greater capabilities while still maintaining simplicity.

/etc/rc.boot or /etc/rc?.d

These directories contain all the files necessary to control system services and configure runlevels. A skeleton file is provided in /etc/init.d/skeleton

/etc/rcS.d

The scripts in this directory are executed once when booting the system, even when booting directly into single user mode. The files are all symbolic links, the real files are located in /etc/init.d/. For a more general discussion of this technique, see /etc/init.d/README.

/etc/resolv.conf

Configuration of how DNS is to occur is defined in this file. It tells the name resolver libraries where they need to go to find information not found in the /etc/hosts file. This always has at least one nameserver line, but preferably three. The resolver uses each in turn. More than the first three can be included but anything beyond the first three will be ignored. Two lines that appear in the /etc/resolv.conf file are domain and search. Both of these are mutually exclusive options, and where both show up, the last one wins. Other entries beyond the three discussed here are listed in the man pages but aren't often used.

/etc/rmt

This is not a mistake. This shell script (/etc/rmt) has been provided for compatibility with other Unix-like systems, some of which have utilities that expect to find (and execute) rmt in the /etc directory on remote systems.

/etc/rpc

The rpc file contains user readable names that can be used in place of rpc program numbers. Each line has the following information: –name of server for the rpc program –rpc program number –aliases Items are separated by any number of blanks and/or tab characters. A `#` indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

```
# /etc/rpc:
# $Id: etc.xml,v 1.10 2004/02/03 21:42:57 binh Exp $
#
# rpc 88/08/01 4.0 RPCSRC; from 1.12    88/02/07 SMI

portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs              100003  nfsprog
ypserv           100004  ypprog
mountd           100005  mount showmount
ypbind           100007
```

Linux Filesystem Hierarchy

```
walld      100008  rwall shutdown
yppasswdd  100009  yppasswd
etherstatd 100010  etherstat
rquotad    100011  rquotaproq quota rquota
sprayd     100012  spray
3270_mapper 100013
rje_mapper  100014
selection_svc 100015  selnsvc
database_svc 100016
rex        100017  rex
alis       100018
sched       100019
llockmgr   100020
nlockmgr   100021
x25.inr     100022
statmon    100023
status      100024
bootparam   100026
ypupdated   100028  ypuupdate
keyserv     100029  keyserver
tfsd        100037
nsed        100038
nsemntd    100039
pcnfsd     150001
amd         300019  amq
sgi_fam    391002
ugidd      545580417
bwnfsd     788585389
```

/etc/samba

Samba configuration files. A 'LanManager' like file and printer server for Unix. The Samba software suite is a collection of programs that implements the SMB protocol for unix systems, allowing you to serve files and printers to Windows, NT, OS/2 and DOS clients. This protocol is sometimes also referred to as the LanManager or NetBIOS protocol.

/etc/sane.d

Sane configuration files. SANE stands for "Scanner Access Now Easy" and is an application programming interface (API) that provides standardized access to any raster image scanner hardware (flatbed scanner, hand-held scanner, video- and still-cameras, frame-grabbers, etc.). The SANE API is public domain and its discussion and development is open to everybody. The current source code is written for UNIX (including GNU/Linux) and is available under the GNU General Public License (the SANE API is available to proprietary applications and backends as well, however).

SANE is a universal scanner interface. The value of such a universal interface is that it allows writing just one driver per image acquisition device rather than one driver for each device and application. So, if you have three applications and four devices, traditionally you'd have had to write 12 different programs. With SANE, this number is reduced to seven: the three applications plus the four drivers. Of course, the savings get even bigger as more and more drivers and/or applications are added.

Not only does SANE reduce development time and code duplication, it also raises the level at which applications can work. As such, it will enable applications that were previously unheard of in the UNIX world. While SANE is primarily targeted at a UNIX environment, the standard has been carefully designed to make it possible to implement the API on virtually any hardware or operating system.

While SANE is an acronym for ``Scanner Access Now Easy'' the hope is of course that SANE is indeed sane in the sense that it will allow easy implementation of the API while accommodating all

Linux Filesystem Hierarchy

features required by today's scanner hardware and applications. Specifically, SANE should be broad enough to accommodate devices such as scanners, digital still and video cameras, as well as virtual devices like image file filters.

If you're familiar with TWAIN, you may wonder why there is a need for SANE. Simply put, TWAIN does not separate the user-interface from the driver of a device. This, unfortunately, makes it difficult, if not impossible, to provide network transparent access to image acquisition devices (which is useful if you have a LAN full of machines, but scanners connected to only one or two machines; it's obviously also useful for remote-controlled cameras and such). It also means that any particular TWAIN driver is pretty much married to a particular GUI API (be it Win32 or the Mac API). In contrast, SANE cleanly separates device controls from their representation in a user-interface. As a result, SANE has no difficulty supporting command-line driven interfaces or network-transparent scanning. For these reasons, it is unlikely that there will ever be a SANE backend that can talk to a TWAIN driver. The converse is no problem though: it would be pretty straight forward to access SANE devices through a TWAIN source. In summary, if TWAIN had been just a little better designed, there would have been no reason for SANE to exist, but things being the way they are, TWAIN simply isn't SANE.

/etc/securetty

Identifies secure terminals, i.e., the terminals from which root is allowed to log in. Typically only the virtual consoles are listed, so that it becomes impossible (or at least harder) to gain superuser privileges by breaking into a system over a modem or a network.

```
# /etc/securetty: list of terminals on which root is allowed to login.
# See securetty(5) and login(1).
console

# Standard consoles
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
tty9
tty10
tty11
tty12

# Same as above, but these only occur with devfs devices
vc/1
vc/2
vc/3
vc/4
vc/5
vc/6
vc/7
vc/8
vc/9
vc/10
vc/11
vc/12
```

/etc/sensors.conf

Linux Filesystem Hierarchy

Configuration file for libsensors. A set of libraries designed to ascertain current hardware states via motherboard sensor chips. Useful statistics such as core voltages, CPU temperature can be determined through third party utilities that make use of these libraries such as 'gkrellm'. If you do not wish to install these packages you may also utilise the /proc filesystem real-time nature.

/etc/sudoers

Sudoers file. This file must be edited with the 'visudo' command as root. The sudo command allows an authenticated user to execute an authorized command as root. Both the effective UID and GID are set to 0 (you are basically root). It determines which users are authorized and which commands they are authorized to use. Configuration of this command is via this file.

/etc/shadow

Shadow password file on systems with shadow password software installed (PAMs). Shadow passwords move the encrypted password from /etc/passwd into /etc/shadow; the latter is not readable by anyone except root. This makes it more difficult to crack passwords.

/etc/shadow--

Old /etc/shadow file.

/etc/sysctl.conf

Configuration file for setting system variables, most notably kernel parameters. 'sysctl' is a means of configuring certain aspects of the kernel at run-time, and the /proc/sys/ directory is there so that you don't even need special tools to do it!

/etc/security

Essential to security. This subdirectory allows administrators to impose quota limits, access limits and also to configure PAM environments.

/etc/serial.conf

Serial port configuration. Changeable parameters include speed, baud rate, port, irq and type.

/etc/services

A definition of the networks, services and the associated port for each protocol that are available on this system. For example, web services (http) are assigned to port 80 by default. # /etc/services: # \$Id: etc.xml,v 1.10 2004/02/03 21:42:57 binh Exp \$ # # Network services, Internet style # # Note that it is presently the policy of IANA to assign a single # well-known port number for both TCP and UDP; hence, most entries # here have two entries even if the protocol doesn't support UDP # operations.

Updated from RFC 1700, ``Assigned Numbers'' (October # 1994). Not all ports are included, only the more common ones. echo 7/tcp echo 7/udp discard 9/tcp sink null discard 9/udp sink null systat 11/tcp users daytime 13/tcp daytime 13/udp netstat 15/tcp qotd 17/tcp quote msp 18/tcp # message send protocol msp 18/udp # message send protocol chargen 19/tcp ttyst source chargen 19/udp ttyst source ftp-data 20/tcp ftp 21/tcp fsp 21/udp fspd ssh 22/tcp # SSH Remote Login Protocol ssh 22/udp # SSH Remote Login Protocol telnet 23/tcp # 24 – private smtp 25/tcp mail # 26 – unassigned time 37/tcp timserver time 37/udp timserver rlp 39/udp resource # resource location nameserver 42/tcp name # IEN 116 whois 43/tcp nickname re-mail-ck 50/tcp # Remote Mail Checking Protocol re-mail-ck 50/udp # Remote Mail Checking Protocol domain 53/tcp nameserver # name-domain server domain 53/udp nameserver netbios-ns 137/tcp # NETBIOS Name Service netbios-ns 137/udp netbios-dgm 138/tcp # NETBIOS Datagram Service netbios-dgm 138/udp netbios-ssn 139/tcp # NETBIOS session service netbios-ssn 139/udp x11 6000/tcp x11-0 # X windows system x11 6000/udp x11-0 # X windows system

/etc/shells

Lists trusted shells. The chsh command allows users to change their login shell only to shells listed in this file. ftpd, the server process that provides FTP services for a machine, will check that the user's shell is listed in /etc/shells and will not let people log in unless the shell is listed there. There are also some display managers that will passively or actively (dependent upon on distribution and display manager being used) refuse a user access to the system unless their shell is one of those listed here.

Linux Filesystem Hierarchy

```
# /etc/shells: valid login shells
/bin/ash
/bin/bash
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/zsh
/bin/sash
/bin/zsh
/usr/bin/esh
```

/etc/skel/

The default files for each new user are stored in this directory. Each time a new user is added, these skeleton files are copied into their home directory. An average system would have: .alias, .bash_profile, .bashrc and .cshrc files. Other files are left up to the system administrator.

/etc/sysconfig/

This directory contains configuration files and subdirectories for the setup of system configuration specifics and for the boot process, like 'clock', which sets the timezone, or 'keyboard' which controls the keyboard map. The contents may vary drastically depending on which distribution and what utilities you have installed. For example, on a Redhat or Mandrake based system it is possible to alter an endless array of attributes from the default desktop to whether DMA should be enabled for your IDE devices. On our Debian reference system though this folder is almost expedient containing only two files hwconf and soundcard which are both configured by the Redhat utilities hwconf and sndconfig respectively.

/etc/slip

Configuration files for the setup and operation of SLIP (serial line IP) interface. Generally unused nowadays. This protocol has been superceded by the faster and more efficient PPP protocol.

/etc/screenrc

This is the system wide screenrc. You can use this file to change the default behavior of screen system wide or copy it to ~/.screenrc and use it as a starting point for your own settings. Commands in this file are used to set options, bind screen functions to keys, redefine terminal capabilities, and to automatically establish one or more windows at the beginning of your screen session. This is not a comprehensive list of options, look at the screen manual for details on everything that you can put in this file.

/etc/scrollkeeper.conf

A free electronic cataloging system for documentation. It stores metadata specified by the <http://www.ibiblio.org/osrt/omf/> (Open Source Metadata Framework) as well as certain metadata extracted directly from documents (such as the table of contents). It provides various functionality pertaining to this metadata to help browsers, such as sorting the registered documents or searching the metadata for documents which satisfy a set of criteria.

/etc/ssh

'ssh' configuration files. 'ssh' is a secure rlogin/rsh/rpc replacement (OpenSSH). This is the portable version of OpenSSH, a free implementation of the Secure Shell protocol as specified by the IETF secsh working group. 'ssh' (Secure Shell) is a program for logging into a remote machine and for executing commands on a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel. It is intended as a replacement for rlogin, rsh and rcp, and can be used to provide applications with a secure communication channel. It should be noted that in some

Linux Filesystem Hierarchy

countries, particularly Iraq, and Pakistan, it may be illegal to use any encryption at all without a special permit.

/etc/syslog.conf

Lists where log files should go, what messages are written to them and the level of verbosity. It is also now possible to filter based on message content, message integrity, message encryption (near future), portability and better network forwarding.

/etc/termcap

The terminal capability database. Describes the "escape sequences" by which various terminals can be controlled. Programs are written so that instead of directly outputting an escape sequence that only works on a particular brand of terminal, they look up the correct sequence to do whatever it is they want to do in /etc/termcap. As a result most programs work with most kinds of terminals.

/etc/timezone

local timezone.

/etc/updatedb.conf

Sets environment variables that are used by updatedb which therefore configures the database for 'locate', a utility that locates a pattern in a database of filenames and returns the filenames that match.

```
# This file sets environment variables which are used by updatedb

# filesystems which are pruned from updatedb database
PRUNEFS="NFS nfs afs proc smbfs autoiso9660 ncpfs coda devpts ftpfs"
export PRUNEFS
# paths which are pruned from updatedb database
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /amd /alex /var/spool"
export PRUNEPATHS
# netpaths which are added
NETPATHS=""
export NETPATHS
```

/etc/vga

The configuration file for the svgalib is stored in this directory. svgalib provides graphics capabilities to programs running on the system console, without going through the X Window System. It uses direct access to the video hardware to provide low-level access to the standard VGA and SVGA graphics modes. It only works with some video hardware; so use with caution.

/etc/vim

Contains configuration files for both vim and its X based counterpart gvim. A wide range of options can be accessed through these two files such as automatic indentation, syntax highlighting, etc....

/etc/xinetd.d/

The original 'inetd' daemon has now been superceded by the much improved 'xinetd'. 'inetd' should be run at boot time by /etc/init.d/inetd (or /etc/rc.local on some systems). It then listens for connections on certain Internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases). Essentially, inetd allows running one daemon to invoke several others, reducing load on the system. Services controlled via xinetd put their configuration files here.

/etc/zlogin

System-wide .zlogin file for zsh(1). This file is sourced only for login shells. It should contain commands that should be executed only in login shells. It should be used to set the terminal type and run a series of external commands (fortune, msgs, from, etc.)

/etc/zlogout

Commands to be executed upon user exit from the zsh. Its control is system-wide but the .zlogout file for zsh(1) does override it in terms of importance.

/etc/zprofile

Linux Filesystem Hierarchy

System-wide .zprofile file for zsh(1). This file is sourced only for login shells (i.e. Shells invoked with “–” as the first character of argv[0], and shells invoked with the -l flag.)

/etc/zshenv

System-wide .zshenv file for zsh(1). This file is sourced on all invocations of the shell. If the -f flag is present or if the NO_RCS option is set within this file, all other initialization files are skipped. This file should contain commands to set the command search path, plus other important environment variables. This file should not contain commands that produce output or assume the shell is attached to a tty.

/etc/zshrc

System-wide .zshrc file for zsh(1). This file is sourced only for interactive shells. It should contain commands to set up aliases, functions, options, key bindings, etc.

Compliance with the FSSTND requires that the following directories, or symbolic links to directories are required in /etc:

```
opt      Configuration for /opt
X11     Configuration for the X Window system (optional)
sgml    Configuration for SGML (optional)
xml     Configuration for XML (optional)
```

The following directories, or symbolic links to directories must be in /etc, if the corresponding subsystem is installed:

```
opt      Configuration for /opt
```

The following files, or symbolic links to files, must be in /etc if the corresponding subsystem is installed (it is recommended that files be stored in subdirectories of /etc/ rather than directly in /etc):

```
csh.login  Systemwide initialization file for C shell logins (optional)
exports    NFS filesystem access control list (optional)
fstab      Static information about filesystems (optional)
ftpusers   FTP daemon user access control list (optional)
gateways   File which lists gateways for routed (optional)
gettydefs  Speed and terminal settings used by getty (optional)
group      User group file (optional)
host.conf   Resolver configuration file (optional)
hosts      Static information about host names (optional)
hosts.allow Host access file for TCP wrappers (optional)
hosts.deny  Host access file for TCP wrappers (optional)
hosts.equiv List of trusted hosts for rlogin, rsh, rcp (optional)
hosts.lpd   List of trusted hosts for lpd (optional)
inetd.conf  Configuration file for inetd (optional)
inittab    Configuration file for init (optional)
issue      Pre-login message and identification file (optional)
ld.so.conf  List of extra directories to search for shared libraries
            (optional)
motd       Post-login message of the day file (optional)
mtab       Dynamic information about filesystems (optional)
mtools.conf Configuration file for mtools (optional)
networks   Static information about network names (optional)
passwd     The password file (optional)
printcap   The lpd printer capability database (optional)
profile    Systemwide initialization file for sh shell logins (optional)
protocols  IP protocol listing (optional)
resolv.conf Resolver configuration file (optional)
rpc        RPC protocol listing (optional)
securetty  TTY access control for root login (optional)
services   Port names for network services (optional)
```

Linux Filesystem Hierarchy

shells Pathnames of valid login shells (optional)
syslog.conf Configuration file for syslogd (optional)

mtab does not fit the static nature of /etc: it is excepted for historical reasons. On some Linux systems, this may be a symbolic link to /proc/mounts, in which case this exception is not required.

/etc/opt : Configuration files for /opt
Host-specific configuration files for add-on application software packages must be installed within the directory /etc/opt/&60;subdir&62;, where &60;subdir&62; is the name of the subtree in /opt where the static data from that package is stored.

No structure is imposed on the internal arrangement of /etc/opt/&60;subdir&62;. If a configuration file must reside in a different location in order for the package or system to function properly, it may be placed in a location other than /etc/opt/&60;subdir&62;.

The rationale behind this subtree is best explained by referring to the rationale for /opt.

/etc/X11 : Configuration for the X Window System (optional)
/etc/X11 is the location for all X11 host-specific configuration. This directory is necessary to allow local control if /usr is mounted read only.

The following files, or symbolic links to files, must be in /etc/X11 if the corresponding subsystem is installed:

Xconfig The configuration file for early versions of XFree86 (optional)
XF86Config The configuration file for XFree86 versions 3 and 4 (optional)
Xmodmap Global X11 keyboard modification file (optional)

Subdirectories of /etc/X11 may include those for xdm and for any other programs (some window managers, for example) that need them.

/etc/X11/xdm holds the configuration files for xdm. These are most of the files previously found in /usr/lib/X11/xdm. Some local variable data for xdm is stored in /var/lib/xdm.

It is recommended that window managers with only one configuration file which is a default .*wmrc file must name it system.*wmrc (unless there is a widely-accepted alternative name) and not use a subdirectory. Any window manager subdirectories must be identically named to the actual window manager binary.

/etc/sgml : Configuration files for SGML (optional)
Generic configuration files defining high-level parameters of the SGML systems are installed here. Files with names *.conf indicate generic configuration files. File with names *.cat are the DTD-specific centralized catalogs, containing references to all other catalogs needed to use the given DTD. The super catalog file catalog references all the centralized catalogs.

/etc/xml : Configuration files for XML (optional)
Generic configuration files defining high-level parameters of the XML systems are installed here. Files with names *.conf indicate generic configuration files. The super catalog file catalog references all the centralized catalogs.

1.7. /home

Linux is a multi-user environment so each user is also assigned a specific directory that is accessible only to them and the system administrator. These are the user home directories, which can be found under '/home/\$USER' (~). It is your playground: everything is at your command, you can write files, delete them, install programs, etc.... Your home directory contains your personal configuration files, the so-called dot files (their name is preceded by a dot). Personal configuration files are usually 'hidden', if you want to see them, you either have to turn on the appropriate option in your file manager or run ls with the -a switch. If there is a conflict between personal and system wide configuration files, the settings in the personal file will prevail.

Dotfiles most likely to be altered by the end user are probably your .xsession and .bashrc files. The configuration files for X and Bash respectively. They allow you to be able to change the window manager to be startup upon login and also aliases, user-specified commands and environment variables respectively. Almost always when a user is created their dotfiles will be taken from the /etc/skel directory where system administrators place a sample file that user's can modify to their hearts content.

/home can get quite large and can be used for storing downloads, compiling, installing and running programs, your mail, your collection of image or sound files etc.

The FSSTND states that:

/home is a fairly standard concept, but it is clearly a site-specific filesystem.

Different people prefer to place user accounts in a variety of places. This section describes only a suggested placement for user home directories; nevertheless we recommend that all FHS-compliant distributions use this as the default location for home directories. On small systems, each user's directory is typically one of the many subdirectories of /home such as /home smith, /home/torvalds, /home/operator, etc. On large systems (especially when the /home directories are shared amongst many hosts using NFS) it is useful to subdivide user home directories. Subdivision may be accomplished by using subdirectories such as /home/staff, /home/guests, /home/students, etc.

The setup will differ from host to host. Therefore, no program should rely on this location.

If you want to find out a user's home directory, you should use the getpwent(3) library function rather than relying on /etc/passwd because user information may be stored remotely using systems such as NIS.

User specific configuration files for applications are stored in the user's home directory in a file that starts with the '.' character (a "dot file"). If an application needs to create more than one dot file then they should be placed in a subdirectory with a name starting with a '.' character, (a "dot directory"). In this case the configuration files should not start with the '.' character.

It is recommended that apart from autosave and lock files programs should refrain from creating non dot files or directories in a home directory without user intervention.

1.8. /initrd

initrd provides the capability to load a RAM disk by the boot loader. This RAM disk can then be mounted as the root file system and programs can be run from it. Afterwards, a new root file system can be mounted from a different device. The previous root (from initrd) is then moved to a directory and can be subsequently unmounted.

initrd is mainly designed to allow system startup to occur in two phases, where the kernel comes up with a minimum set of compiled-in drivers, and where additional modules are loaded from initrd.

Operation

When using initrd, the system typically boots as follows:

- 1) the boot loader loads the kernel and the initial RAM disk
- 2) the kernel converts initrd into a "normal" RAM disk and frees the memory used by initrd
- 3) initrd is mounted read-write as root
- 4) /linuxrc is executed (this can be any valid executable, including shell scripts; it is run with uid 0 and can do basically everything init can do)
- 5) linuxrc mounts the "real" root file system
- 6) linuxrc places the root file system at the root directory using the pivot_root system call
- 7) the usual boot sequence (e.g. invocation of /sbin/init) is performed on the root file system
- 8) the initrd file system is removed

Note that changing the root directory does not involve unmounting it. It is therefore possible to leave processes running on initrd during that procedure. Also note that file systems mounted under initrd continue to be accessible.

Usage scenarios

The main motivation for implementing initrd was to allow for modular kernel configuration at system installation.

The procedure would work as follows:

- 1) system boots from floppy or other media with a minimal kernel (e.g. support for RAM disks, initrd, a.out, and the Ext2 FS) and loads initrd
- 2) /linuxrc determines what is needed to (1) mount the "real" root FS (i.e. device type, device drivers, file system) and (2) the distribution media (e.g. CD-ROM, network, tape, ...). This can be done by asking the user, by auto-probing, or by using a hybrid approach.
- 3) /linuxrc loads the necessary kernel modules
- 4) /linuxrc creates and populates the root file system (this doesn't have to be a very usable system yet)
- 5) /linuxrc invokes pivot_root to change the root file system and execs - via chroot - a program that continues the installation
- 6) the boot loader is installed
- 7) the boot loader is configured to load an initrd with the set of modules that was used to bring up the system (e.g. /initrd can

Linux Filesystem Hierarchy

```
be modified, then unmounted, and finally, the image is written
from /dev/ram0 or /dev/rd/0 to a file)
8) now the system is bootable and additional installation tasks
can be performed
```

The key role of initrd here is to re-use the configuration data during normal system operation without requiring the use of a bloated "generic" kernel or re-compiling or re-linking the kernel.

A second scenario is for installations where Linux runs on systems with different hardware configurations in a single administrative domain. In such cases, it is desirable to generate only a small set of kernels (ideally only one) and to keep the system-specific part of configuration information as small as possible. In this case, a common initrd could be generated with all the necessary modules. Then, only /linuxrc or a file read by it would have to be different.

A third scenario are more convenient recovery disks, because information like the location of the root FS partition doesn't have to be provided at boot time, but the system loaded from initrd can invoke a user-friendly dialog and it can also perform some sanity checks (or even some form of auto-detection).

Last not least, CD-ROM distributors may use it for better installation from CD, e.g. by using a boot floppy and bootstrapping a bigger RAM disk via initrd from CD; or by booting via a loader like LOADLIN or directly from the CD-ROM, and loading the RAM disk from CD without need of floppies.

1.9. /lib

The /lib directory contains kernel modules and those shared library images (the C programming code library) needed to boot the system and run the commands in the root filesystem, ie. by binaries in /bin and /sbin. Libraries are readily identifiable through their filename extension of *.so. Windows equivalent to a shared library would be a DLL (dynamically linked library) file. They are essential for basic system functionality. Kernel modules (drivers) are in the subdirectory /lib/modules/'kernel-version'. To ensure proper module compilation you should ensure that /lib/modules/'kernel-version'/kernel/build points to /usr/src/'kernel-version' or ensure that the Makefile knows where the kernel source itself are located.

/lib/'machine-architecture'

Contains platform/architecture dependent libraries.

/lib/iptables

iptables shared library files.

/lib/kbd

Contains various keymaps.

/lib/modules/'kernel-version'

The home of all the kernel modules. The organisation of files here is reasonably clear so no requires no elaboration.

/lib/modules/'kernel-version'/isapnppmap.dep

has details on ISA based cards, the modules that they require and various other attributes.

/lib/modules/'kernel-version'/modules.dep

lists all modules dependencies. This file can be updated using the depmod command.

/lib/modules/'kernel-version'/pcimap

is the PCI equivalent of the /lib/modules/'kernel-version'/isapnppmap.dep file.

/lib/modules/'kernel-version'/usbmap

is the USB equivalent of the /lib/modules/'kernel-version'/isapnppmap.dep file.

Linux Filesystem Hierarchy

/lib/oss

All OSS (Open Sound System) files are installed here by default.

/lib/security

PAM library files.

The FSSTND states that the /lib directory contains those shared library images needed to boot the system and run the commands in the root filesystem, ie. by binaries in /bin and /sbin.

Shared libraries that are only necessary for binaries in /usr (such as any X Window binaries) must not be in /lib. Only the shared libraries required to run binaries in /bin and /sbin may be here. In particular, the library libm.so.* may also be placed in /usr/lib if it is not required by anything in /bin or /sbin.

At least one of each of the following filename patterns are required (they may be files, or symbolic links):

```
libc.so.* The dynamically-linked C library (optional)
ld*       The execution time linker/loader (optional)
```

If a C preprocessor is installed, /lib/cpp must be a reference to it, for historical reasons. The usual placement of this binary is /usr/bin/cpp.

The following directories, or symbolic links to directories, must be in /lib, if the corresponding subsystem is installed:

```
modules Loadable kernel modules (optional)

/lib<qual> : Alternate format essential shared libraries (optional)
```

There may be one or more variants of the /lib directory on systems which support more than one binary format requiring separate libraries.

This is commonly used for 64-bit or 32-bit support on systems which support multiple binary formats, but require libraries of the same name. In this case, /lib32 and /lib64 might be the library directories, and /lib a symlink to one of them.

If one or more of these directories exist, the requirements for their contents are the same as the normal /lib directory, except that /lib<qual>/cpp is not required.

/lib<qual>/cpp is still permitted: this allows the case where /lib and /lib<qual> are the same (one is a symbolic link to the other).

1.10. /lost+found

As was explained earlier during the overview of the FSSTND, Linux should always go through a proper shutdown. Sometimes your system might crash or a power failure might take the machine down. Either way, at the next boot, a lengthy filesystem check (the speed of this check is dependent on the type of filesystem that you actually use. ie. ext3 is faster than ext2 because it is a journalled filesystem) using fsck will be done. Fsck will go through the system and try to recover any corrupt files that it finds. The result of this recovery operation will be placed in this directory. The files recovered are not likely to be complete or make much sense but there always is a chance that something worthwhile is recovered. Each partition has its own lost+found directory. If you find files in there, try to move them back to their original location. If you find something like a broken symbolic link to 'file', you have to reinstall the file/s from the corresponding RPM,

Linux Filesystem Hierarchy

since your file system got damaged so badly that the files were mutilated beyond recognition. Below is an example of a /lost+found directory. As you can see, the vast majority of files contained here are in actual fact sockets. As for the rest of the other files they were found to be damaged system files and personal files. These files were not able to be recovered.

```
total 368
-rw-r--r-- 1 root root 110891 Oct 5 14:14 #388200
-rw-r--r-- 1 root root 215 Oct 5 14:14 #388201
-rw-r--r-- 1 root root 110303 Oct 6 23:09 #388813
-rw-r--r-- 1 root root 141 Oct 6 23:09 #388814
-rw-r--r-- 1 root root 110604 Oct 6 23:09 #388815a
-rw-r--r-- 1 root root 194 Oct 6 23:09 #388816
srwxr-xr-x 1 root root 0 Oct 6 13:00 #51430
srwxr-xr-x 1 root root 0 Oct 6 00:23 #51433
-rw----- 1 root root 63 Oct 6 00:23 #51434
srwxr-xr-x 1 root root 0 Oct 6 13:00 #51436
srwxrwxrwx 1 root root 0 Oct 6 00:23 #51437
srwx----- 1 root root 0 Oct 6 00:23 #51438
-rw----- 1 root root 63 Oct 6 13:00 #51439
srwxrwxrwx 1 root root 0 Oct 6 13:00 #51440
srwx----- 1 root root 0 Oct 6 13:00 #51442
-rw----- 1 root root 63 Oct 6 23:09 #51443
srwx----- 1 root root 0 Oct 6 10:40 #51445
srwxrwxrwx 1 root root 0 Oct 6 23:09 #51446
srwx----- 1 root root 0 Oct 6 23:09 #51448
```

1.11. /media

Amid much controversy and consternation on the part of system and network administrators a directory containing mount points for removable media has now been created. Funnily enough, it has been named /media.

This directory contains subdirectories which are used as mount points for removable media such as floppy disks, cdroms and zip disks.

The motivation for the creation of this directory has been that historically there have been a number of other different places used to mount removable media such as /cdrom, /mnt or /mnt/cdrom. Placing the mount points for all removable media directly in the root directory would potentially result in a large number of extra directories in /. Although the use of subdirectories in /mnt as a mount point has recently been common, it conflicts with a much older tradition of using /mnt directly as a temporary mount point.

The following directories, or symbolic links to directories, must be in /media, if the corresponding subsystem is installed:

```
floppy      Floppy drive (optional)
cdrom       CD-ROM drive (optional)
cdrecorder  CD writer (optional)
zip         Zip drive (optional)
```

On systems where more than one device exists for mounting a certain type of media, mount directories can be created by appending a digit to the name of those available above starting with '0', but the unqualified name must also exist.

A compliant implementation with two CDROM drives might have /media/cdrom0

and /media/cdrom1 with /media/cdrom a symlink to either of these.

Please see the section on the /mnt directory to achieve a better understanding of the process on mounting and unmounting filesystems.

1.12. /mnt

This is a generic mount point under which you mount your filesystems or devices. Mounting is the process by which you make a filesystem available to the system. After mounting your files will be accessible under the mount-point. This directory usually contains mount points or sub-directories where you mount your floppy and your CD. You can also create additional mount-points here if you wish. Standard mount points would include /mnt/cdrom and /mnt/floppy. There is no limitation to creating a mount-point anywhere on your system but by convention and for sheer practicality do not litter your file system with mount-points. It should be noted that some distributions like Debian allocate /floppy and /cdrom as mount points while Redhat and Mandrake puts them in /mnt/floppy and /mnt/cdrom respectively.

However, it should be noted that as of FSSTND version 2.3 the purpose of this directory has changed.

This directory is provided so that the system administrator may temporarily mount a filesystem as needed. The content of this directory is a local issue and should not affect the manner in which any program is run.

This directory must not be used by installation programs: a suitable temporary directory not in use by the system must be used instead.

1.12.1. Mounting and unmounting

Before one can use a filesystem, it has to be *mounted*. The operating system then does various bookkeeping things to make sure that everything works. Since all files in UNIX are in a single directory tree, the mount operation will make it look like the contents of the new filesystem are the contents of an existing subdirectory in some already mounted filesystem.

The mounts could be done as in the following example:

```
$ mount /dev/hda2 /home  
$ mount /dev/hda3 /usr  
$
```

The **mount** command takes two arguments. The first one is the device file corresponding to the disk or partition containing the filesystem. The second one is the directory below which it will be mounted. After these commands the contents of the two filesystems look just like the contents of the /home and /usr directories, respectively. One would then say that `/dev/hda2 is mounted on /home`, and similarly for /usr. To look at either filesystem, one would look at the contents of the directory on which it has been mounted, just as if it were any other directory. Note the difference between the device file, /dev/hda2, and the mounted-on directory, /home. The device file gives access to the raw contents of the disk, the mounted-on directory gives access to the files on the disk. The mounted-on directory is called the *mount point*.

Linux supports many filesystem types. **mount** tries to guess the type of the filesystem. You can also use the -t fstype option to specify the type directly; this is sometimes necessary, since the heuristics **mount** uses do not always work. For example, to mount an MS-DOS floppy, you could use the following command:

Linux Filesystem Hierarchy

```
$ mount -t msdos /dev/fd0 /floppy  
$
```

The mounted-on directory need not be empty, although it must exist. Any files in it, however, will be inaccessible by name while the filesystem is mounted. (Any files that have already been opened will still be accessible. Files that have hard links from other directories can be accessed using those names.) There is no harm done with this, and it can even be useful. For instance, some people like to have `/tmp` and `/var/tmp` synonymous, and make `/tmp` be a symbolic link to `/var/tmp`. When the system is booted, before the `/var` filesystem is mounted, a `/var/tmp` directory residing on the root filesystem is used instead. When `/var` is mounted, it will make the `/var/tmp` directory on the root filesystem inaccessible. If `/var/tmp` didn't exist on the root filesystem, it would be impossible to use temporary files before mounting `/var`.

If you don't intend to write anything to the filesystem, use the `-r` switch for **mount** to do a *read-only mount*. This will make the kernel stop any attempts at writing to the filesystem, and will also stop the kernel from updating file access times in the inodes. Read-only mounts are necessary for unwritable media, e.g., CD-ROMs.

The alert reader has already noticed a slight logistical problem. How is the first filesystem (called the *root filesystem*, because it contains the root directory) mounted, since it obviously can't be mounted on another filesystem? Well, the answer is that it is done by magic.

For more information, see the kernel source or the Kernel Hackers' Guide.

The root filesystem is magically mounted at boot time, and one can rely on it to always be mounted. If the root filesystem can't be mounted, the system does not boot. The name of the filesystem that is magically mounted as root is either compiled into the kernel, or set using LILO or **rdev**.

The root filesystem is usually first mounted read-only. The startup scripts will then run **fsck** to verify its validity, and if there are no problems, they will *re-mount* it so that writes will also be allowed. **fsck** must not be run on a mounted filesystem, since any changes to the filesystem while **fsck** is running *will* cause trouble. Since the root filesystem is mounted read-only while it is being checked, **fsck** can fix any problems without worry, since the remount operation will flush any metadata that the filesystem keeps in memory.

On many systems there are other filesystems that should also be mounted automatically at boot time. These are specified in the `/etc/fstab` file; see the `fstab` man page for details on the format. The details of exactly when the extra filesystems are mounted depend on many factors, and can be configured by each administrator if need be.

When a filesystem no longer needs to be mounted, it can be unmounted with **umount**.

It should of course be **umount**, but the `n` mysteriously disappeared in the 70s, and hasn't been seen since. Please return it to Bell Labs, NJ, if you find it.

umount takes one argument: either the device file or the mount point. For example, to unmount the directories of the previous example, one could use the commands

```
$ umount /dev/hda2  
$ umount /usr  
$
```

See the man page for further instructions on how to use the command. It is imperative that you always unmount a mounted floppy. *Don't just pop the floppy out of the drive!* Because of disk caching, the data is not

Linux Filesystem Hierarchy

necessarily written to the floppy until you unmount it, so removing the floppy from the drive too early might cause the contents to become garbled. If you only read from the floppy, this is not very likely, but if you write, even accidentally, the result may be catastrophic.

Mounting and unmounting requires super user privileges, i.e., only root can do it. The reason for this is that if any user can mount a floppy on any directory, then it is rather easy to create a floppy with, say, a Trojan horse disguised as `/bin/sh`, or any other often used program. However, it is often necessary to allow users to use floppies, and there are several ways to do this:

- Give the users the root password. This is obviously bad security, but is the easiest solution. It works well if there is no need for security anyway, which is the case on many non-networked, personal systems.
- Use a program such as **sudo** to allow users to use `mount`. This is still bad security, but doesn't directly give super user privileges to everyone. [1]
- Make the users use **mtools**, a package for manipulating MS-DOS filesystems, without mounting them. This works well if MS-DOS floppies are all that is needed, but is rather awkward otherwise.
- List the floppy devices and their allowable mount points together with the suitable options in `/etc/fstab`.

The last alternative can be implemented by adding a line like the following to the `/etc/fstab` file:

```
/dev/fd0 /floppy  
msdos user,noauto 0 0
```

The columns are: device file to mount, directory to mount on, filesystem type, options, backup frequency (used by **dump**), and **fsck** pass number (to specify the order in which filesystems should be checked upon boot; 0 means no check).

The `noauto` option stops this mount to be done automatically when the system is started (i.e., it stops **mount -a** from mounting it). The `user` option allows any user to mount the filesystem, and, because of security reasons, disallows execution of programs (normal or setuid) and interpretation of device files from the mounted filesystem. After this, any user can mount a floppy with an msdos filesystem with the following command:

```
$ mount /floppy  
$
```

The floppy can (and needs to, of course) be unmounted with the corresponding **umount** command.

If you want to provide access to several types of floppies, you need to give several mount points. The settings can be different for each mount point. For example, to give access to both MS-DOS and ext2 floppies, you could have the following two lines in `/etc/fstab`:

```
/dev/fd0 /dosfloppy msdos user,noauto 0 0 /dev/fd0  
/ext2floppy ext2 user,noauto 0 0
```

For MS-DOS filesystems (not just floppies), you probably want to restrict access to it by using the `uid`, `gid`, and `umask` filesystem options, described in detail on the **mount** manual page. If you aren't careful, mounting an MS-DOS filesystem gives everyone at least read access to the files in it, which is not a good idea.

1.13. /opt

This directory is reserved for all the software and add-on packages that are not part of the default installation. For example, StarOffice, Kylix, Netscape Communicator and WordPerfect packages are normally found here. To comply with the FSSTND, all third party applications should be installed in this directory. Any package to be installed here must locate its static files (ie. extra fonts, clipart, database files) must locate its static files in a separate /opt/'package' or /opt/'provider' directory tree (similar to the way in which Windows will install new software to its own directory tree C:\Windows\Program Files\"Program Name"), where 'package' is a name that describes the software package and 'provider' is the provider's LANANA registered name.

Although most distributions neglect to create the directories /opt/bin, /opt/doc, /opt/include, /opt/info, /opt/lib, and /opt/man they are reserved for local system administrator use. Packages may provide "front-end" files intended to be placed in (by linking or copying) these reserved directories by the system administrator, but must function normally in the absence of these reserved directories. Programs to be invoked by users are located in the directory /opt/'package'/bin. If the package includes UNIX manual pages, they are located in /opt/'package'/man and the same substructure as /usr/share/man must be used. Package files that are variable must be installed in /var/opt. Host-specific configuration files are installed in /etc/opt.

Under no circumstances are other package files to exist outside the /opt, /var/opt, and /etc/opt hierarchies except for those package files that must reside in specific locations within the filesystem tree in order to function properly. For example, device lock files in /var/lock and devices in /dev. Distributions may install software in /opt, but must not modify or delete software installed by the local system administrator without the assent of the local system administrator.

The use of /opt for add-on software is a well-established practice in the UNIX community. The System V Application Binary Interface [AT&T 1990], based on the System V Interface Definition (Third Edition) and the Intel Binary Compatibility Standard v. 2 (iBCS2) provides for an /opt structure very similar to the one defined here.

Generally, all data required to support a package on a system must be present within /opt/'package', including files intended to be copied into /etc/opt/'package' and /var/opt/'package' as well as reserved directories in /opt. The minor restrictions on distributions using /opt are necessary because conflicts are possible between distribution installed and locally installed software, especially in the case of fixed pathnames found in some binary software.

The structure of the directories below /opt/'provider' is left up to the packager of the software, though it is recommended that packages are installed in /opt/'provider'/'package' and follow a similar structure to the guidelines for /opt/package. A valid reason for diverging from this structure is for support packages which may have files installed in /opt/ 'provider'/lib or /opt/'provider'/bin.

1.14. /proc

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.

Linux Filesystem Hierarchy

The most distinctive thing about files in this directory is the fact that all of them have a file size of 0, with the exception of kcore, mtrr and self. A directory listing looks similar to the following:

```
total 525256
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 1
dr-xr-xr-x  3 daemon    root          0 Jan 19 15:00 109
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 170
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 173
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 178
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 2
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 3
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 4
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 421
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 425
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 433
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 439
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 444
dr-xr-xr-x  3 daemon    daemon        0 Jan 19 15:00 446
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 449
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 453
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 456
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 458
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 462
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 463
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 464
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 465
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 466
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 467
dr-xr-xr-x  3 gdm       gdm          0 Jan 19 15:00 472
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 483
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 5
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 6
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 7
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 8
-r--r--r--   1 root      root          0 Jan 19 15:00 apm
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 bus
-r--r--r--   1 root      root          0 Jan 19 15:00 cmdline
-r--r--r--   1 root      root          0 Jan 19 15:00 cpuinfo
-r--r--r--   1 root      root          0 Jan 19 15:00 devices
-r--r--r--   1 root      root          0 Jan 19 15:00 dma
dr-xr-xr-x  3 root      root          0 Jan 19 15:00 driver
-r--r--r--   1 root      root          0 Jan 19 15:00 execdomains
-r--r--r--   1 root      root          0 Jan 19 15:00 fb
-r--r--r--   1 root      root          0 Jan 19 15:00 filesystems
dr-xr-xr-x  2 root      root          0 Jan 19 15:00 fs
dr-xr-xr-x  4 root      root          0 Jan 19 15:00 ide
-r--r--r--   1 root      root          0 Jan 19 15:00 interrupts
-r--r--r--   1 root      root          0 Jan 19 15:00 iomem
-r--r--r--   1 root      root          0 Jan 19 15:00 ioports
dr-xr-xr-x  18 root     root          0 Jan 19 15:00 irq
-r-----   1 root      root          536809472 Jan 19 15:00 kcore
-r-----   1 root      root          0 Jan 19 14:58 kms
-r--r--r--   1 root      root          0 Jan 19 15:00 ksyms
-r--r--r--   1 root      root          0 Jan 19 15:00 loadavg
-r--r--r--   1 root      root          0 Jan 19 15:00 locks
-r--r--r--   1 root      root          0 Jan 19 15:00 mdstat
-r--r--r--   1 root      root          0 Jan 19 15:00 meminfo
-r--r--r--   1 root      root          0 Jan 19 15:00 misc
-r--r--r--   1 root      root          0 Jan 19 15:00 modules
-r--r--r--   1 root      root          0 Jan 19 15:00 mounts
-rw-r--r--  1 root      root          137 Jan 19 14:59 mtrr
```

Linux Filesystem Hierarchy

```
dr-xr-xr-x    3 root    root      0 Jan 19 15:00 net
dr-xr-xr-x    2 root    root      0 Jan 19 15:00 nv
-r--r--r--    1 root    root      0 Jan 19 15:00 partitions
-r--r--r--    1 root    root      0 Jan 19 15:00 pci
dr-xr-xr-x    4 root    root      0 Jan 19 15:00 scsi
lrwxrwxrwx    1 root    root      64 Jan 19 14:58 self -> 483
-rw-r--r--    1 root    root      0 Jan 19 15:00 slabinfo
-r--r--r--    1 root    root      0 Jan 19 15:00 stat
-r--r--r--    1 root    root      0 Jan 19 15:00 swaps
dr-xr-xr-x   10 root    root      0 Jan 19 15:00 sys
dr-xr-xr-x    2 root    root      0 Jan 19 15:00 sysvipc
dr-xr-xr-x    4 root    root      0 Jan 19 15:00 tty
-r--r--r--    1 root    root      0 Jan 19 15:00 uptime
-r--r--r--    1 root    root      0 Jan 19 15:00 version
```

Each of the numbered directories corresponds to an actual process ID. Looking at the process table, you can match processes with the associated process ID. For example, the process table might indicate the following for the secure shell server:

```
# ps ax | grep sshd
439 ? S 0:00 /usr/sbin/sshd
```

Details of this process can be obtained by looking at the associated files in the directory for this process, /proc/460. You might wonder how you can see details of a process that has a file size of 0. It makes more sense if you think of it as a window into the kernel. The file doesn't actually contain any data; it just acts as a pointer to where the actual process information resides. For example, a listing of the files in the /proc/460 directory looks similar to the following:

```
total 0
-r--r--r--    1 root    root      0 Jan 19 15:02 cmdline
lrwxrwxrwx    1 root    root      0 Jan 19 15:02 cwd -> /
-r-----    1 root    root      0 Jan 19 15:02 environ
lrwxrwxrwx    1 root    root      0 Jan 19 15:02 exe -> /usr/sbin/sshd
dr-x-----    2 root    root      0 Jan 19 15:02 fd
-r--r--r--    1 root    root      0 Jan 19 15:02 maps
-rw-----    1 root    root      0 Jan 19 15:02 mem
lrwxrwxrwx    1 root    root      0 Jan 19 15:02 root -> /
-r--r--r--    1 root    root      0 Jan 19 15:02 stat
-r--r--r--    1 root    root      0 Jan 19 15:02 statm
-r--r--r--    1 root    root      0 Jan 19 15:02 status
```

The purpose and contents of each of these files is explained below:

/proc/PID/cmdline

Command line arguments.

/proc/PID/cpu

Current and last cpu in which it was executed.

/proc/PID/cwd

Link to the current working directory.

/proc/PID/environ

Values of environment variables.

/proc/PID/exe

Link to the executable of this process.

/proc/PID/fd

Directory, which contains all file descriptors.

/proc/PID/maps

Memory maps to executables and library files.

/proc/PID/mem

Linux Filesystem Hierarchy

Memory held by this process.

/proc/PID/root

Link to the root directory of this process.

/proc/PID/stat

Process status.

/proc/PID/statm

Process memory status information.

/proc/PID/status

Process status in human readable form.

Should you wish to know more, the man page for proc describes each of the files associated with a running process ID in far greater detail.

Even though files appear to be of size 0, examining their contents reveals otherwise:

```
# cat status
```

```
Name: sshd
State: S (sleeping)
Tgid: 439
Pid: 439
PPid: 1
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 32
Groups:
VmSize:      2788 kB
VmLck:        0 kB
VmRSS:       1280 kB
VmData:       252 kB
VmStk:        16 kB
VmExe:        268 kB
VmLib:       2132 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000001000
SigCgt: 00000000000014005
CapInh: 0000000000000000
CapPrm: 00000000fffffeff
CapEff: 00000000fffffeff
```

The files in the /proc directory act very similar to the process ID subdirectory files. For example, examining the contents of the /proc/interrupts file displays something like the following:

```
# cat interrupts
```

```
          CPU0
0:    32657      XT-PIC  timer
1:    1063       XT-PIC  keyboard
2:      0        XT-PIC  cascade
8:      3        XT-PIC  rtc
9:      0        XT-PIC  cmpci
11:   332       XT-PIC  nvidia
14:   5289      XT-PIC  ide0
15:   13        XT-PIC  ide1
NMI:    0
ERR:    0
```

Linux Filesystem Hierarchy

Each of the numbers down the left-hand column represents the interrupt that is in use. Examining the contents of the file dynamically gathers the associated data and displays it to the screen. Most of the /proc file system is read-only; however, some files allow kernel variable to be changed. This provides a mechanism to actually tune the kernel without recompiling and rebooting.

The procinfo utility summarizes /proc file system information into a display similar to the following:

```
# /usr/bin/procinfo

Linux 2.4.18 (root@DEB) (gcc 2.95.4 20011002 ) #2 1CPU [DEB. (none) ]

Memory:      Total        Used        Free        Shared       Buffers       Cached
Mem:        513908     107404    406504          0        2832     82180
Swap:       265032          0    265032

Bootup: Sun Jan 19 14:58:27 2003      Load average: 0.29 0.13 0.05 1/30 566

user :      0:00:10.26   2.3%  page in :    74545  disk 1:      6459r      796w
nice :      0:00:00.00   0.0%  page out:    9416  disk 2:      19r       0w
system:     0:00:19.55   4.5%  swap in :        1
idle :      0:06:48.30  93.2%  swap out:        0
uptime:     0:07:18.11           context :   22059

irq  0:    43811 timer                  irq  9:        0 cmpci
irq  1:    1427 keyboard                irq 11:     332 nvidia
irq  2:        0 cascade [4]            irq 12:        2
irq  6:        2                         irq 14:    7251 ide0
irq  8:        3 rtc                     irq 15:     83 ide1
```

/proc/apm

Advanced power management info.

/proc/bus

Directory containing bus specific information.

/proc/cmdline

Kernel command line.

/proc/cpuinfo

Information about the processor, such as its type, make, model, and performance.

/proc/devices

List of device drivers configured into the currently running kernel (block and character).

/proc/dma

Shows which DMA channels are being used at the moment.

/proc/driver

Various drivers grouped here, currently rtc

/proc/execdomains

Execdomains, related to security.

/proc/fb

Frame Buffer devices.

/proc/filesystems

Filesystems configured/supported into/by the kernel.

/proc/fs

File system parameters, currently nfs/exports.

/proc/ide

This subdirectory contains information about all IDE devices of which the kernel is aware. There is one subdirectory for each IDE controller, the file drivers and a link for each IDE device, pointing to

Linux Filesystem Hierarchy

the device directory in the controller-specific subtree. The file drivers contains general information about the drivers used for the IDE devices. More detailed information can be found in the controller-specific subdirectories. These are named ide0, ide1 and so on. Each of these directories contains the files shown here:

```
/proc/ide/ide?/channel
    IDE channel (0 or 1)
/proc/ide/ide?/config
    Configuration (only for PCI/IDE bridge)
/proc/ide/ide?/mate
    Mate name (onchip partnered controller)
/proc/ide/ide?/model
    Type/Chipset of IDE controller
    Each device connected to a controller has a separate subdirectory in the controllers directory.
    The following files listed are contained in these directories:
/proc/ide/ide?/model/cache
    The cache.
/proc/ide/ide?/model/capacity
    Capacity of the medium (in 512Byte blocks)
/proc/ide/ide?/model/driver
    driver and version
/proc/ide/ide?/model/geometry
    physical and logical geometry
/proc/ide/ide?/model/identify
    device identify block
/proc/ide/ide?/model/media
    media type
/proc/ide/ide?/model/model
    device identifier
/proc/ide/ide?/model/settings
    device setup
/proc/ide/ide?/model/smart_thresholds
    IDE disk management thresholds
/proc/ide/ide?/model/smart_values
    IDE disk management values
```

/proc/interrupts

Shows which interrupts are in use, and how many of each there have been.

You can, for example, check which interrupts are currently in use and what they are used for by looking in the file /proc/interrupts:

```
# cat /proc/interrupts

CPU0 0: 8728810
XT-PIC timer 1: 895
XT-PIC keyboard 2:
0 XT-PIC cascade 3: 531695
XT-PIC aha152x 4: 2014133
XT-PIC serial 5: 44401
XT-PIC pcnet_cs 8: 2
XT-PIC rtc 11: 8
XT-PIC i82365 12: 182918
XT-PIC PS/2 Mouse 13: 1
XT-PIC fpu 14: 1232265
```

Linux Filesystem Hierarchy

```
XT-PIC ide0 15: 7  
XT-PIC ide1 NMI: 0
```

In 2.4 based kernels a couple of lines were added to this file LOC & ERR (this is the output of an SMP machine):

```
# cat /proc/interrupts
```

```
CPU0 CPU1  
0: 1243498 1214548 IO-APIC-edge timer  
1: 8949 8958 IO-APIC-edge keyboard  
2: 0 0 XT-PIC cascade  
5: 11286 10161 IO-APIC-edge soundblaster  
8: 1 0 IO-APIC-edge rtc  
9: 27422 27407 IO-APIC-edge 3c503  
12: 113645 113873 IO-APIC-edge PS/2 Mouse  
13: 0 0 XT-PIC fpu 14: 22491 24012 IO-APIC-edge ide0  
15: 2183 2415 IO-APIC-edge ide1  
17: 30564 30414 IO-APIC-level eth0  
18: 177 164 IO-APIC-level bttv NMI: 2457961 2457959  
LOC: 2457882 2457881 ERR: 2155
```

NMI is incremented in this case because every timer interrupt generates a NMI (Non Maskable Interrupt) which is used by the NMI Watchdog to detect lookups.

LOC is the local interrupt counter of the internal APIC of every CPU.

ERR is incremented in the case of errors in the IO-APIC bus (the bus that connects the CPUs in an SMP system). This means that an error has been detected, the IO-APIC automatically retries the transmission, so it should not be a big problem, but you should read the SMP-FAQ.

In this context it could be interesting to note the new irq directory in 2.4. It could be used to set IRQ to CPU affinity, this means that you can "hook" an IRQ to only one CPU, or to exclude a CPU from handling IRQs. The contents of the irq subdir is one subdir for each IRQ, and one file; prof_cpu_mask. For example,

```
# ls /proc/irq/ 0 10 12 14 16 18 2 4 6 8 prof_cpu_mask  
1 11 13 15 17 19 3 5 7 9
```

```
# ls /proc/irq/0/ smp_affinity
```

The contents of the prof_cpu_mask file and each smp_affinity file for each IRQ is the same by default:

```
# cat /proc/irq/0/smp_affinity  
ffffffffff
```

It's a bitmask, in which you can specify which CPUs can handle the IRQ, you can set it by doing:

```
# echo 1 > /proc/irq/prof_cpu_mask
```

This means that only the first CPU will handle the IRQ, but you can also echo 5 which means that only the first and fourth CPU can handle the IRQ. The way IRQs are routed is handled by the IO-APIC, and its Round Robin between all the CPUs which are allowed to handle it. As usual the kernel has more info than you and does a better job than you, so the defaults are the

Linux Filesystem Hierarchy

best choice for almost everyone.

/proc/iomem

Memory map.

/proc/ioports

Which I/O ports are in use at the moment.

/proc/irq

Masks for irq to cpu affinity.

/proc/isapnp

ISA PnP (Plug&Play) Info.

/proc/kcore

An image of the physical memory of the system (can be ELF or A.OUT (deprecated in 2.4)). This is exactly the same size as your physical memory, but does not really take up that much memory; it is generated on the fly as programs access it. (Remember: unless you copy it elsewhere, nothing under /proc takes up any disk space at all.)

/proc/kmsg

Messages output by the kernel. These are also routed to syslog.

/proc/ksyms

Kernel symbol table.

/proc/loadavg

The 'load average' of the system; three indicators of how much work the system has done during the last 1, 5 & 15 minutes.

/proc/locks

Kernel locks.

/proc/meminfo

Information about memory usage, both physical and swap. Concatenating this file produces similar results to using 'free' or the first few lines of 'top'.

/proc/misc

Miscellaneous pieces of information. This is for information that has no real place within the rest of the proc filesystem.

/proc/modules

Kernel modules currently loaded. Typically its output is the same as that given by the 'lsmod' command.

/proc/mounts

Mounted filesystems

/proc/mtrr

Information regarding mtrrs. (On Intel P6 family processors (Pentium Pro, Pentium II and later) the Memory Type Range Registers (MTRRs) may be used to control processor access to memory ranges. This is most useful when you have a video (VGA) card on a PCI or AGP bus. Enabling write-combining allows bus write transfers to be combined into a larger transfer before bursting over the PCI/AGP bus. This can increase performance of image write operations 2.5 times or more. The Cyrix 6x86, 6x86MX and M II processors have Address Range Registers (ARRs) which provide a similar functionality to MTRRs. For these, the ARRs are used to emulate the MTRRs. The AMD K6-2 (stepping 8 and above) and K6-3 processors have two MTRRs. These are supported. The AMD Athlon family provide 8 Intel style MTRRs. The Centaur C6 (WinChip) has 8 MCRs, allowing write-combining. These are also supported. The VIA Cyrix III and VIA C3 CPUs offer 8 Intel style MTRRs.) For more details regarding mtrr technology see /usr/src/linux/Documentation/mtrr.txt.

/proc/net

Status information about network protocols.

IPv6 information

/proc/net/udp6

Linux Filesystem Hierarchy

- UDP sockets (IPv6).
- /proc/net/tcp6
 - TCP sockets (IPv6).
- /proc/net/raw6
 - Raw device statistics (IPv6).
- /proc/net/igmp6
 - IP multicast addresses, which this host joined (IPv6).
- /proc/net/if_inet6
 - List of IPv6 interface addresses.
- /proc/net/ipv6_route
 - Kernel routing table for IPv6.
- /proc/net/rt6_stats
 - Global IPv6 routing tables statistics.
- /proc/net/sockstat6
 - Socket statistics (IPv6).
- /proc/net/snmp6
 - Snmp data (IPv6).

General Network information

- /proc/net/arp
 - Kernel ARP table.
- /proc/net/dev
 - network devices with statistics.
- /proc/net/dev_mcast
 - the Layer2 multicast groups which a device is listening to (interface index, label, number of references, number of bound addresses).
- /proc/net/dev_stat
 - network device status.
- /proc/net/ip_fwchains
 - Firewall chain linkage.
- /proc/net/ip_fwnames
 - Firewall chain names.
- /proc/net/ip_masq
 - Directory containing the masquerading tables.
- /proc/net/ip_masquerade
 - Major masquerading table.
- /proc/net/netstat
 - Network statistics.
- /proc/net/raw
 - raw device statistics.
- /proc/net/route
 - Kernel routing table.
- /proc/net/rpc
 - Directory containing rpc info.
- /proc/net/rt_cache
 - Routing cache.
- /proc/net/snmp
 - SNMP data.
- /proc/net/sockstat
 - Socket statistics.
- /proc/net/tcp

Linux Filesystem Hierarchy

TCP sockets.

/proc/net/tr_rif
Token ring RIF routing table.

/proc/net/udp
UDP sockets.

/proc/net/unix
UNIX domain sockets.

/proc/net/wireless
Wireless interface data (Wavelan etc).

/proc/net/igmp
IP multicast addresses, which this host joined.

/proc/net/psched
Global packet scheduler parameters.

/proc/net/netlink
List of PF_NETLINK sockets.

/proc/net/ip_mr_vifs
List of multicast virtual interfaces.

/proc/net/ip_mr_cache
List of multicast routing cache.

You can use this information to see which network devices are available in your system and how much traffic was routed over those devices. In addition, each Channel Bond interface has its own directory. For example, the bond0 device will have a directory called /proc/net/bond0/. It will contain information that is specific to that bond, such as the current slaves of the bond, the link status of the slaves, and how many times the slaves link has failed.

/proc/parport
The directory /proc/parport contains information about the parallel ports of your system. It has one subdirectory for each port, named after the port number (0,1,2,...).

/proc/parport/autoprobe
Any IEEE-1284 device ID information that has been acquired.

/proc/parport/devices
list of the device drivers using that port. A + will appear by the name of the device currently using the port (it might not appear against any).

/proc/parport/hardware
Parallel port's base address, IRQ line and DMA channel.

/proc/parport/irq
IRQ that parport is using for that port. This is in a separate file to allow you to alter it by writing a new value in (IRQ number or none).

/proc/partitions
Table of partitions known to the system

/proc/pci, /proc/bus/pci
Deprecated info of PCI bus.

/proc/rtc
Real time clock

/proc/scsi
If you have a SCSI host adapter in your system, you'll find a subdirectory named after the driver for this adapter in /proc/scsi. You'll also see a list of all recognized SCSI devices in /proc/scsi. The directory named after the driver has one file for each adapter found in the system. These files contain information about the controller, including the used IRQ and the IO address range. The amount of information shown is dependent on the adapter you use.

Linux Filesystem Hierarchy

/proc/self

A symbolic link to the process directory of the program that is looking at /proc. When two processes look at /proc, they get different links. This is mainly a convenience to make it easier for programs to get at their process directory.

/proc/slabinfo

The slabinfo file gives information about memory usage at the slab level. Linux uses slab pools for memory management above page level in version 2.2. Commonly used objects have their own slab pool (such as network buffers, directory cache, and so on).

/proc/stat

Overall/various statistics about the system, such as the number of page faults since the system was booted.

/proc/swaps

Swap space utilization

/proc/sys

This is not only a source of information, it also allows you to change parameters within the kernel without the need for recompilation or even a system reboot. Take care when attempting this as it can both optimize your system and also crash it. It is advisable to read both documentation and source before actually making adjustments. The entries in /proc may change slightly between kernel versions, so if there is any doubt review the kernel documentation in the directory

/usr/src/linux/Documentation. Under some circumstances, you may have no alternative but to reboot the machine once an error occurs. To change a value, simply echo the new value into the file. An example is given below in the section on the file system data. Of course, you need to be 'root' to do any of this. You can create your own boot script to perform this every time your system boots.

/proc/sys/fs

Contains file system data. This subdirectory contains specific file system, file handle, inode, dentry and quota information.

dentry-state

Status of the directory cache. Since directory entries are dynamically allocated and deallocated, this file indicates the current status. It holds six values, in which the last two are not used and are always zero. The others are listed below:

File	Content
nr_dentry	Almost always zero
nr_unused	Number of unused cache entries
age_limit	in seconds after the entry may be reclaimed, when memory is short want_pages internally

dquot-max

The file dquot-max shows the maximum number of cached disk quota entries.

dquot-nr

shows the number of allocated disk quota entries and the number of free disk quota entries. If the number of available cached disk quotas is very low and you have a large number of simultaneous system users, you might want to raise the limit.

file-nr and file-max

The kernel allocates file handles dynamically, but doesn't free them again at this time. The value in file-max denotes the maximum number of file handles that the Linux kernel will allocate. When you get a lot of error messages about running out of file handles, you might want to raise this limit. The default value is 4096. To change it, just write the new number into the file:

Linux Filesystem Hierarchy

```
# cat /proc/sys/fs/file-max
4096
# echo 8192 > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

This method of revision is useful for all customizable parameters of the kernel – simply echo the new value to the corresponding file.

The three values in file-nr denote the number of allocated file handles, the number of used file handles, and the maximum number of file handles. When the allocated file handles come close to the maximum, but the number of actually used handles is far behind, you've encountered a peak in your usage of file handles and you don't need to increase the maximum. inode-state, inode-nr and inode-max

As with file handles, the kernel allocates the inode structures dynamically, but can't free them yet.

The value in inode-max denotes the maximum number of inode handlers. This value should be 3 to 4 times larger than the value in file-max, since stdin, stdout, and network sockets also need an inode struct to handle them. If you regularly run out of inodes, you should increase this value.

The file inode-nr contains the first two items from inode-state, so we'll skip to that file... inode-state contains three actual numbers and four dummy values. The numbers are nr_inodes, nr_free_inodes, and preshrink (in order of appearance).

nr_inodes

Denotes the number of inodes the system has allocated. This can be slightly more than inode-max because Linux allocates them one pageful at a time.

nr_free_inodes

Represents the number of free inodes and preshrink is nonzero when nr_inodes is greater than inode-max and the system needs to prune the inode list instead of allocating more.

super-nr and super-max

Again, super block structures are allocated by the kernel, but not freed. The file super-max contains the maximum number of super block handlers, where super-nr shows the number of currently allocated ones. Every mounted file system needs a super block, so if you plan to mount lots of file systems, you may want to increase these numbers.

binfmt_misc

This handles the kernel support for miscellaneous binary formats. binfmt_misc provides the ability to register additional binary formats to the kernel without compiling an additional module/kernel. Therefore, binfmt_misc needs to know magic numbers at the beginning or the filename extension of the binary. It works by maintaining a linked list of structs that contain a description of a binary format, including a magic with size (or the filename extension), offset and mask, and the interpreter name. On request it invokes the given interpreter with the original program as argument, as binfmt_java and binfmt_em86 and binfmt_mz do. Since binfmt_misc does not define any default binary-formats, you have to register an additional binary-format. There are two general files in binfmt_misc and one file per registered format. The two general files are register and status. To register a new binary format you have to issue the command echo :name:type:offset:magic:mask:interpreter: > /proc/sys/fs/binfmt_misc/register with appropriate name (the name for the /proc-dir entry), offset (defaults to 0, if omitted), magic, mask (which can be omitted, defaults to all 0xff) and last but not least, the interpreter that is to be invoked (for example and testing /bin/echo).

Type can be M for usual magic matching or E for filename extension matching (give extension in place of magic). If you do a cat on the file /proc/sys/fs/binfmt_misc/status, you will get the current status (enabled/disabled) of binfmt_misc. Change the status by echoing 0 (disables) or 1 (enables) or -1 (caution: this clears all previously registered binary formats) to

Linux Filesystem Hierarchy

status. For example echo 0 > status to disable binfmt_misc (temporarily). Each registered handler has an entry in /proc/sys/fs/binfmt_misc. These files perform the same function as status, but their scope is limited to the actual binary format. By 'cating' this file, you also receive all related information about the interpreter/magic of the binfmt. An example of the usage of binfmt_misc (emulate binfmt_java) follows:

```
cd /proc/sys/fs/binfmt_misc
echo ':Java:M::\xca\xfe\xba\xbe:::/usr/local/java/bin/javawrapper:' > register
echo ':HTML:E::html:::/usr/local/java/bin/appletviewer:' > register
echo ':Applet:M::<!--applet:::/usr/local/java/bin/appletviewer:' > register
echo ':DExE:M::\x0eDEX:::/usr/bin/dosexec:' < register
```

These four lines add support for Java executables and Java applets (like binfmt_java, additionally recognizing the .html extension with no need to put <!--applet> to every applet file). You have to install the JDK and the shell-script /usr/local/java/bin/javawrapper too. It works around the brokenness of the Java filename handling. To add a Java binary, just create a link to the class-file somewhere in the path.

/proc/sys/kernel

This directory reflects general kernel behaviors and the contents will be dependent upon your configuration. Here you'll find the most important files, along with descriptions of what they mean and how to use them.

/proc/sys/kernel/acct

The file contains three values; highwater, lowwater, and frequency. It exists only when BSD-style process accounting is enabled. These values control its behavior. If the free space on the file system where the log lives goes below lowwater percentage, accounting suspends. If it goes above highwater percentage, accounting resumes. Frequency determines how often you check the amount of free space (value is in seconds). Default settings are: 4, 2, and 30. That is, suspend accounting if there is less than 2 percent free; resume it if we have a value of 3 or more percent; consider information about the amount of free space valid for 30 seconds

/proc/sys/kernel/ctrl-alt-del

When the value in this file is 0, ctrl-alt-del is trapped and sent to the init program to handle a graceful restart. However, when the value is greater than zero, Linux's reaction to this key combination will be an immediate reboot, without syncing its dirty buffers. It should be noted that when a program (like dosemu) has the keyboard in raw mode, the ctrl-alt-del is intercepted by the program before it ever reaches the kernel tty layer, and it is up to the program to decide what to do with it.

/proc/sys/kernel/domainname, /proc/sys/kernel/hostname

These files can be controlled to set the NIS domainname and hostname of your box. For the classic darkstar.frop.org a simple: # echo "darkstar" > /proc/sys/kernel/hostname # echo "frop.org" > /proc/sys/kernel/domainname would suffice to set your hostname and NIS domainname. /proc/sys/kernel/osrelease, /proc/sys/kernel/ostype, /proc/sys/kernel/version The names make it pretty obvious what these fields contain: # cat /proc/sys/kernel/osrelease 2.2.12 # cat /proc/sys/kernel/ostype Linux # cat /proc/sys/kernel/version #4 Fri Oct 1 12:41:14 PDT 1999 The files osrelease and ostype should be clear enough. Version needs a little more clarification. The #4 means that this is the 4th kernel built from this source base and the date after it indicates the time the kernel was built. The only way to tune these values is to rebuild the kernel.

/proc/sys/kernel/panic

Linux Filesystem Hierarchy

The value in this file represents the number of seconds the kernel waits before rebooting on a panic. When you use the software watchdog, the recommended setting is 60. If set to 0, the auto reboot after a kernel panic is disabled, which is the default setting.

/proc/sys/kernel/printk

The four values in printk denote * console_loglevel, * default_message_loglevel, * minimum_console_level and * default_console_loglevel respectively. These values influence printk() behavior when printing or logging error messages, which come from inside the kernel. See syslog(2) for more information on the different log levels.

/proc/sys/kernel/console_loglevel

Messages with a higher priority than this will be printed to the console.

/proc/sys/kernel/default_message_level

Messages without an explicit priority will be printed with this priority.

/proc/sys/kernel/minimum_console_loglevel

Minimum (highest) value to which the console_loglevel can be set.

/proc/sys/kernel/default_console_loglevel

Default value for console_loglevel.

/proc/sys/kernel/sg-big-buff

This file shows the size of the generic SCSI (sg) buffer. At this point, you can't tune it yet, but you can change it at compile time by editing include/scsi/sg.h and changing the value of SG_BIG_BUFF. If you use a scanner with SANE (Scanner Access Now Easy) you might want to set this to a higher value. Refer to the SANE documentation on this issue.

/proc/sys/kernel/modprobe

The location where the modprobe binary is located. The kernel uses this program to load modules on demand.

/proc/sys/vm

The files in this directory can be used to tune the operation of the virtual memory (VM) subsystem of the Linux kernel. In addition, one of the files (bdflush) has some influence on disk usage.

nfract

This parameter governs the maximum number of dirty buffers in the buffer cache. Dirty means that the contents of the buffer still have to be written to disk (as opposed to a clean buffer, which can just be forgotten about). Setting this to a higher value means that Linux can delay disk writes for a long time, but it also means that it will have to do a lot of I/O at once when memory becomes short. A lower value will spread out disk I/O more evenly.

ndirty

Ndirty gives the maximum number of dirty buffers that bdflush can write to the disk at one time. A high value will mean delayed, bursty I/O, while a small value can lead to memory shortage when bdflush isn't woken up often enough.

nrefill

This is the number of buffers that bdflush will add to the list of free buffers when refill_freelist() is called. It is necessary to allocate free buffers beforehand, since the buffers are often different sizes than the memory pages and some bookkeeping needs to be done beforehand. The higher the number, the more memory will be wasted and the less often refill_freelist() will need to run.

nref_dirt

When refill_freelist() comes across more than nref_dirt dirty buffers, it will wake up bdflush.

age_buffer, age_super

Finally, the age_buffer and age_super parameters govern the maximum time Linux waits before writing out a dirty buffer to disk. The value is expressed in jiffies (clockticks), the number of jiffies per second is 100. Age_buffer is the maximum age for data blocks, while age_super is for filesystems meta data.

buffermem

Linux Filesystem Hierarchy

The three values in this file control how much memory should be used for buffer memory. The percentage is calculated as a percentage of total system memory.

The values are:

`min_percent`

This is the minimum percentage of memory that should be spent on buffer memory.

`borrow_percent`

When Linux is short on memory, and the buffer cache uses more than it has been allotted, the memory management (MM) subsystem will prune the buffer cache more heavily than other memory to compensate.

`max_percent`

This is the maximum amount of memory that can be used for buffer memory.

`freepages`

This file contains three values: min, low and high:

`min`

When the number of free pages in the system reaches this number, only the kernel can allocate more memory.

`low`

If the number of free pages falls below this point, the kernel starts swapping aggressively.

`high`

The kernel tries to keep up to this amount of memory free; if memory falls below this point, the kernel starts gently swapping in the hopes that it never has to do really aggressive swapping.

`kswapd`

Kswapd is the kernel swap out daemon. That is, kswapd is that piece of the kernel that frees memory when it gets fragmented or full. Since every system is different, you'll probably want some control over this piece of the system.

The file contains three numbers:

`tries_base`

The maximum number of pages kswapd tries to free in one round is calculated from this number. Usually this number will be divided by 4 or 8 (see mm/vmscan.c), so it isn't as big as it looks. When you need to increase the bandwidth to/from swap, you'll want to increase this number.

`tries_min`

This is the minimum number of times kswapd tries to free a page each time it is called.

Basically it's just there to make sure that kswapd frees some pages even when it's being called with minimum priority.

`swap_cluster`

This is probably the greatest influence on system performance. `swap_cluster` is the number of pages kswapd writes in one turn. You'll want this value to be large so that kswapd does its I/O in large chunks and the disk doesn't have to seek as often, but you don't want it to be too large since that would flood the request queue.

`overcommit_memory`

This file contains one value. The following algorithm is used to decide if there's enough memory: if the value of `overcommit_memory` is positive, then there's always enough memory. This is a useful feature, since programs often malloc() huge amounts of memory 'just in case', while they only use a small part of it. Leaving this value at 0 will lead to the failure of such a huge malloc(), when in fact the system has enough memory for the program to run. On the other hand, enabling this feature can cause you to run out of memory and thrash the system to death, so large and/or important servers will want to set this value to 0.

`pagecache`

Linux Filesystem Hierarchy

This file does exactly the same job as buffermem, only this file controls the amount of memory allowed for memory mapping and generic caching of files. You don't want the minimum level to be too low, otherwise your system might thrash when memory is tight or fragmentation is high.

pagetable_cache

The kernel keeps a number of page tables in a per-processor cache (this helps a lot on SMP systems). The cache size for each processor will be between the low and the high value. On a low-memory, single CPU system, you can safely set these values to 0 so you don't waste memory. It is used on SMP systems so that the system can perform fast pagetable allocations without having to acquire the kernel memory lock. For large systems, the settings are probably fine. For normal systems they won't hurt a bit. For small systems (less than 16MB ram) it might be advantageous to set both values to 0.

swapctl

This file contains no less than 8 variables. All of these values are used by kswapd. The first four variables sc_max_page_age, sc_page_advance, sc_page_decline and sc_page_initial_age are used to keep track of Linux's page aging. Page ageing is a bookkeeping method to track which pages of memory are often used, and which pages can be swapped out without consequences.

When a page is swapped in, it starts at sc_page_initial_age (default 3) and when the page is scanned by kswapd, its age is adjusted according to the following scheme.

If the page was used since the last time we scanned, its age is increased by sc_page_advance (default 3). Where the maximum value is given by sc_max_page_age (default 20). Otherwise (meaning it wasn't used) its age is decreased by sc_page_decline (default 1).

When a page reaches age 0, it's ready to be swapped out.

The variables sc_age_cluster_fract, sc_age_cluster_min, sc_pageout_weight and sc_bufferout_weight, can be used to control kswapd's aggressiveness in swapping out pages. Sc_age_cluster_fract is used to calculate how many pages from a process are to be scanned by kswapd. The formula used is

(sc_age_cluster_fract divided by 1024) times resident set size

So if you want kswapd to scan the whole process, sc_age_cluster_fract needs to have a value of 1024. The minimum number of pages kswapd will scan is represented by sc_age_cluster_min, which is done so that kswapd will also scan small processes. The values of sc_pageout_weight and sc_bufferout_weight are used to control how many tries kswapd will make in order to swap out one page/buffer. These values can be used to fine-tune the ratio between user pages and buffer/cache memory. When you find that your Linux system is swapping out too many process pages in order to satisfy buffer memory demands, you may want to either increase sc_bufferout_weight, or decrease the value of sc_pageout_weight.

/proc/sys/dev

Device specific parameters. Currently there is only support for CDROM drives, and for those, there is only one read-only file containing information about the CD-ROM drives attached to the system:
>cat /proc/sys/dev/cdrom/info
CD-ROM information, Id: cdrom.c 2.55 1999/04/25 drive name: sr0
hdb drive speed: 32 40 drive # of slots: 1 0 Can close tray: 1 1 Can open tray: 1 1 Can lock tray: 1 1
Can change speed: 1 1 Can select disk: 0 1 Can read multisession: 1 1 Can read MCN: 1 1 Reports media changed: 1 1 Can play audio: 1 1 You see two drives, sr0 and hdb, along with a list of their features.

SUNRPC

/proc/sys/sunrpc

This directory contains four files, which enable or disable debugging for the RPC functions NFS, NFS-daemon, RPC and NLM. The default values are 0. They can be set to one to turn debugging on. (The default value is 0 for each)

/proc/sys/net

Linux Filesystem Hierarchy

The interface to the networking parts of the kernel is located in /proc/sys/net. The following table shows all possible subdirectories. You may see only some of them, depending on your kernel's configuration. Our main focus will be on IP networking since AX15, X.25, and DEC Net are only minor players in the Linux world. Should you wish review the online documentation and the kernel source to get a detailed view of the parameters for those protocols not covered here. In this section we'll discuss the subdirectories listed above. As default values are suitable for most needs, there is no need to change these values.

GENERAL PARAMETERS

/proc/sys/net/core	Network core options
rmem_default	The default setting of the socket receive buffer in bytes.
rmem_max	The maximum receive socket buffer size in bytes.
wmem_default	The default setting (in bytes) of the socket send buffer.
wmem_max	The maximum send socket buffer size in bytes.
message_burst and message_cost	These parameters are used to limit the warning messages written to the kernel log from the networking code. They enforce a rate limit to make a denial-of-service attack impossible. A higher message_cost factor, results in fewer messages that will be written. Message_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.
netdev_max_backlog	Maximum number of packets, queued on the INPUT side, when the interface receives packets faster than kernel can process them.
optmem_max	Maximum ancillary buffer size allowed per socket. Ancillary data is a sequence of struct cmsghdr structures with appended data.

UNIX DOMAIN SOCKETS

/proc/sys/net/unix	Parameters for Unix domain sockets
	There are only two files in this subdirectory. They control the delays for deleting and destroying socket descriptors.

IPv4

/proc/sys/net/ipv4	IPV4 settings. IP version 4 is still the most used protocol in Unix networking. It will be replaced by IP version 6 in the next couple of years, but for the moment it's the de facto standard for the internet and is used in most networking environments around the world. Because of the importance of this protocol, we'll have a deeper look into the subtree controlling the behavior of the Ipv4 subsystem of the Linux kernel.
	Let's start with the entries in /proc/sys/net/ipv4.

ICMP settings

icmp_echo_ignore_all and icmp_echo_ignore_broadcasts	Turn on (1) or off (0), if the kernel should ignore all ICMP ECHO requests, or just those to broadcast and multicast addresses.
--	---

Linux Filesystem Hierarchy

Please note that if you accept ICMP echo requests with a broadcast/multi\–cast destination address your network may be used as an exploder for denial of service packet flooding attacks to other hosts.

icmp_destunreach_rate, icmp_echo_reply_rate, icmp_paramprob_rate and icmp_timeexceed_rate

Sets limits for sending ICMP packets to specific targets. A value of zero disables all limiting. Any positive value sets the maximum package rate in hundredth of a second (on Intel systems).

IP settings

ip_autoconfig

This file contains the number one if the host received its IP configuration by RARP, BOOTP, DHCP or a similar mechanism. Otherwise it is zero.

ip_default_ttl

TTL (Time To Live) for IPv4 interfaces. This is simply the maximum number of hops a packet may travel.

ip_dynaddr

Enable dynamic socket address rewriting on interface address change. This is useful for dialup interface with changing IP addresses.

ip_forward

Enable or disable forwarding of IP packages between interfaces. Changing this value resets all other parameters to their default values. They differ if the kernel is configured as host or router.

ip_local_port_range

Range of ports used by TCP and UDP to choose the local port. Contains two numbers, the first number is the lowest port, the second number the highest local port. Default is 1024–4999. Should be changed to 32768–61000 for high–usage systems.

ip_no_pmtu_disc

Global switch to turn path MTU discovery off. It can also be set on a per socket basis by the applications or on a per route basis.

ip_masq_debug

Enable/disable debugging of IP masquerading.

IP fragmentation settings

ipfrag_high_trash and ipfrag_low_trash

Maximum memory used to reassemble IP fragments. When ipfrag_high_trash bytes of memory is allocated for this purpose, the fragment handler will toss packets until ipfrag_low_trash is reached.

ipfrag_time

Time in seconds to keep an IP fragment in memory.

TCP settings

tcp_ecn

This file controls the use of the ECN bit in the IPv4 headers, this is a new feature about Explicit Congestion Notification, but some routers and firewalls block traffic that has this bit set, so it could be necessary to echo 0 to /proc/sys/net/ipv4/tcp_ecn, if you want to talk to this sites. For more info you could read RFC2481.

tcp_retrans_collapse

Bug-to-bug compatibility with some broken printers. On retransmit, try to send larger packets to work around bugs in certain TCP stacks. Can be turned off by setting it to zero.

tcp_keepalive_probes

Number of keep alive probes TCP sends out, until it decides that the connection is broken.

Linux Filesystem Hierarchy

tcp_keepalive_time

How often TCP sends out keep alive messages, when keep alive is enabled. The default is 2 hours.

tcp_syn_retries

Number of times initial SYNs for a TCP connection attempt will be retransmitted. Should not be higher than 255. This is only the timeout for outgoing connections, for incoming connections the number of retransmits is defined by tcp_retries1.

tcp_sack

Enable select acknowledgments after RFC2018.

tcp_timestamps

Enable timestamps as defined in RFC1323.

tcp_stdurg

Enable the strict RFC793 interpretation of the TCP urgent pointer field. The default is to use the BSD compatible interpretation of the urgent pointer pointing to the first byte after the urgent data. The RFC793 interpretation is to have it point to the last byte of urgent data.

Enabling this option may lead to interoperability problems. Disabled by default.

tcp_syncookies

Only valid when the kernel was compiled with CONFIG_SYNCOOKIES. Send out syncookies when the syn backlog queue of a socket overflows. This is to ward off the common 'syn flood attack'. Disabled by default. Note that the concept of a socket backlog is abandoned. This means the peer may not receive reliable error messages from an over loaded server with syncookies enabled.

tcp_window_scaling

Enable window scaling as defined in RFC1323.

tcp_fin_timeout

The length of time in seconds it takes to receive a final FIN before the socket is always closed. This is strictly a violation of the TCP specification, but required to prevent denial-of-service attacks.

tcp_max_ka_probes

Indicates how many keep alive probes are sent per slow timer run. Should not be set too high to prevent bursts.

tcp_max_syn_backlog

Length of the per socket backlog queue. Since Linux 2.2 the backlog specified in listen(2) only specifies the length of the backlog queue of already established sockets. When more connection requests arrive Linux starts to drop packets. When syncookies are enabled the packets are still answered and the maximum queue is effectively ignored.

tcp_retries1

Defines how often an answer to a TCP connection request is retransmitted before giving up.

tcp_retries2

Defines how often a TCP packet is retransmitted before giving up.

/proc/sys/net/ipv4/conf

Here you'll find one subdirectory for each interface the system knows about and one directory called all. Changes in the all subdirectory affect all interfaces, whereas changes in the other subdirectories affect only one interface. All directories have the same entries:

accept_redirects

This switch decides if the kernel accepts ICMP redirect messages or not. The default is 'yes' if the kernel is configured for a regular host and 'no' for a router configuration.

accept_source_route

Should source routed packages be accepted or declined. The default is dependent on the kernel configuration. It's 'yes' for routers and 'no' for hosts.

bootp_relay

Linux Filesystem Hierarchy

Accept packets with source address 0.b.c.d with destinations not to this host as local ones. It is supposed that a BOOTP relay daemon will catch and forward such packets. The default is 0.

forwarding

Enable or disable IP forwarding on this interface.

log_martians

Log packets with source addresses with no known route to kernel log.

mc_forwarding

Do multicast routing. The kernel needs to be compiled with CONFIG_MROUTE and a multicast routing daemon is required.

proxy_arp

Does (1) or does not (0) perform proxy ARP.

rp_filter

Integer value determines if a source validation should be made. 1 means yes, 0 means no.

Disabled by default, but local/broadcast address spoofing is always on. If you set this to 1 on a router that is the only connection for a network to the net, it will prevent spoofing attacks against your internal networks (external addresses can still be spoofed), without the need for additional firewall rules.

secure_redirects

Accept ICMP redirect messages only for gateways, listed in default gateway list. Enabled by default.

shared_media

If it is not set the kernel does not assume that different subnets on this device can communicate directly. Default setting is 'yes'.

send_redirects

Determines whether to send ICMP redirects to other hosts.

Routing settings

The directory /proc/sys/net/ipv4/route contains several file to control routing issues.

error_burst and error_cost

These parameters are used to limit the warning messages written to the kernel log from the routing code. The higher the error_cost factor is, the fewer messages will be written.

Error_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.

flush

Writing to this file results in a flush of the routing cache.

gc_elastic, gc_interval, gc_min_interval, gc_tresh, gc_timeout

Values to control the frequency and behavior of the garbage collection algorithm for the routing cache.

max_size

Maximum size of the routing cache. Old entries will be purged once the cache reached has this size.

max_delay, min_delay

Delays for flushing the routing cache.

redirect_load, redirect_number

Factors which determine if more ICPM redirects should be sent to a specific host. No redirects will be sent once the load limit or the maximum number of redirects has been reached.

redirect_silence

Timeout for redirects. After this period redirects will be sent again, even if this has been stopped, because the load or number limit has been reached.

/proc/sys/net/ipv4/neigh

Linux Filesystem Hierarchy

Network Neighbor handling. It contains settings about how to handle connections with direct neighbors (nodes attached to the same link). As we saw it in the conf directory, there is a default subdirectory which holds the default values, and one directory for each interface. The contents of the directories are identical, with the single exception that the default settings contain additional options to set garbage collection parameters.

In the interface directories you'll find the following entries:

`base_reachable_time`

A base value used for computing the random reachable time value as specified in RFC2461.
`retrans_time`

The time, expressed in jiffies (1/100 sec), between retransmitted Neighbor Solicitation messages. Used for address resolution and to determine if a neighbor is unreachable.

`unres_qlen`

Maximum queue length for a pending arp request – the number of packets which are accepted from other layers while the ARP address is still resolved.

`anycast_delay`

Maximum for random delay of answers to neighbor solicitation messages in jiffies (1/100 sec). Not yet implemented (Linux does not have anycast support yet).

`ucast_solicit`

Maximum number of retries for unicast solicitation.

`mcast_solicit`

Maximum number of retries for multicast solicitation.

`delay_first_probe_time`

Delay for the first time probe if the neighbor is reachable. (see `gc_stale_time`)

`locktime`

An ARP/neighbor entry is only replaced with a new one if the old is at least locktime old.

This prevents ARP cache thrashing.

`proxy_delay`

Maximum time (real time is random [0..proxftime]) before answering to an ARP request for which we have an proxy ARP entry. In some cases, this is used to prevent network flooding.

`proxy_qlen`

Maximum queue length of the delayed proxy arp timer. (see `proxy_delay`).

`app_solicit`

Determines the number of requests to send to the user level ARP daemon. Use 0 to turn off.

`gc_stale_time`

Determines how often to check for stale ARP entries. After an ARP entry is stale it will be resolved again (which is useful when an IP address migrates to another machine). When `ucast_solicit` is greater than 0 it first tries to send an ARP packet directly to the known host. When that fails and `mcast_solicit` is greater than 0, an ARP request is broadcasted.

APPLETALK

`/proc/sys/net/appletalk`

Holds the Appletalk configuration data when Appletalk is loaded. The configurable parameters are:

`aarp-expiry-time`

The amount of time we keep an ARP entry before expiring it. Used to age out old hosts.

`aarp-resolve-time`

The amount of time we will spend trying to resolve an Appletalk address.

`aarp-retransmit-limit`

The number of times we will retransmit a query before giving up.

`aarp-tick-time`

Controls the rate at which expires are checked.

Linux Filesystem Hierarchy

/proc/net/appletalk

Holds the list of active Appletalk sockets on a machine. The fields indicate the DDP type, the local address (in network:node format) the remote address, the size of the transmit pending queue, the size of the received queue (bytes waiting for applications to read) the state and the uid owning the socket.

/proc/net/atalk_iface

lists all the interfaces configured for appletalk. It shows the name of the interface, its Appletalk address, the network range on that address (or network number for phase 1 networks), and the status of the interface.

/proc/net/atalk_route

lists each known network route. It lists the target (network) that the route leads to, the router (may be directly connected), the route flags, and the device the route is using.

IPX

The IPX protocol has no tunable values in proc/sys/net, it does, however, provide proc/net/ipx. This lists each IPX socket giving the local and remote addresses in Novell format (that is network:node:port). In accordance with the strange Novell tradition, everything but the port is in hex. Not_Connected is displayed for sockets that are not tied to a specific remote address. The Tx and Rx queue sizes indicate the number of bytes pending for transmission and reception. The state indicates the state the socket is in and the uid is the owning uid of the socket.

ipx_interface

Lists all IPX interfaces. For each interface it gives the network number, the node number, and indicates if the network is the primary network. It also indicates which device it is bound to (or Internal for internal networks) and the Frame Type if appropriate. Linux supports 802.3, 802.2, 802.2 SNAP and DIX (Blue Book) ethernet framing for IPX.

ipx_route

Table holding a list of IPX routes. For each route it gives the destination network, the router node (or Directly) and the network address of the router (or Connected) for internal networks.

/proc/sysvipc

Info of SysVIPC Resources (msg, sem, shm) (2.4)

/proc/tty

Information about the available and actually used tty's can be found in the directory /proc/tty. You'll find entries for drivers and line disciplines in this directory.

/proc/tty/drivers

list of drivers and their usage.

/proc/tty/ldiscs

registered line disciplines.

/proc/tty/serial

usage statistic and status of single tty lines.

To see which tty's are currently in use, you can simply look into the file /proc/tty/drivers:

```
# cat /proc/tty/drivers
serial          /dev/cua      5  64-127 serial:callout
serial          /dev/ttys     4  64-127 serial
pty_slave       /dev/pts     143 0-255 pty:slave
pty_master      /dev/ptm     135 0-255 pty:master
pty_slave       /dev/pts     142 0-255 pty:slave
pty_master      /dev/ptm     134 0-255 pty:master
pty_slave       /dev/pts     141 0-255 pty:slave
pty_master      /dev/ptm     133 0-255 pty:master
pty_slave       /dev/pts     140 0-255 pty:slave
pty_master      /dev/ptm     132 0-255 pty:master
pty_slave       /dev/pts     139 0-255 pty:slave
```

Linux Filesystem Hierarchy

pty_master	/dev/ptm	131	0-255	pty:master
pty_slave	/dev/pts	138	0-255	pty:slave
pty_master	/dev/ptm	130	0-255	pty:master
pty_slave	/dev/pts	137	0-255	pty:slave
pty_master	/dev/ptm	129	0-255	pty:master
pty_slave	/dev/pts	136	0-255	pty:slave
pty_master	/dev/ptm	128	0-255	pty:master
pty_slave	/dev/ttyp	3	0-255	pty:slave
pty_master	/dev/pty	2	0-255	pty:master
/dev/vc/0	/dev/vc/0	4	0	system:vtmaster
/dev/ptmx	/dev/ptmx	5	2	system
/dev/console	/dev/console	5	1	system:console
/dev/tty	/dev/tty	5	0	system:/dev/tty
unknown	/dev/vc/%d	4	1-63	console

Note that while the above files tend to be easily readable text files, they can sometimes be formatted in a way that is not easily digestible. There are many commands that do little more than read the above files and format them for easier understanding. For example, the free program reads /proc/meminfo and converts the amounts given in bytes to kilobytes (and adds a little more information, as well).

/proc/uptime

The time the system has been up.

/proc/version

The kernel version.

/proc/video

BTTV info of video resources.

1.15. /root

This is the home directory of the System Administrator, 'root'. This may be somewhat confusing ('root on root') but in former days, '/' was root's home directory (hence the name of the Administrator account). To keep things tidier, 'root' got his own home directory. Why not in '/home'? Because '/home' is often located on a different partition or even on another system and would thus be inaccessible to 'root' when – for some reason – only '/' is mounted.

The FSSTND merely states that this is the recommended location for the home directory of 'root'. It is left up to the end user to determine the home directory of 'root'. However, the FSSTND also says that:

If the home directory of the root account is not stored on the root partition it will be necessary to make certain it will default to / if it can not be located.

We recommend against using the root account for tasks that can be performed as an unprivileged user, and that it be used solely for system administration. For this reason, we recommend that subdirectories for mail and other applications not appear in the root account's home directory, and that mail for administration roles such as root, postmaster, and webmaster be forwarded to an appropriate user.

1.16. /sbin

Linux discriminates between 'normal' executables and those used for system maintenance and/or administrative tasks. The latter reside either here or – the less important ones – in /usr/sbin. Locally installed system administration programs should be placed into /usr/local/sbin.

Programs executed after /usr is known to be mounted (when there are no problems) are generally placed into /usr/sbin. This directory contains binaries that are essential to the working of the system. These include system administration as well as maintenance and hardware configuration programs. You may find lilo, fdisk, init, ifconfig, etc.... here.

Another directory that contains system binaries is /usr/sbin. This directory contains other binaries of use to the system administrator. This is where you will find the network daemons for your system along with other binaries that (generally) only the system administrator has access to, but which are not required for system maintenance and repair. Normally, these directories are never part of normal user's \$PATHs, only of roots (PATH is an environment variable that controls the sequence of locations that the system will attempt to look in for commands).

The FSSTND states that:

```
/sbin should contain only binaries essential for booting, restoring,
recovering, and/or repairing the system in addition to the binaries
in /bin.
```

A particular eccentricity of the Linux filesystem hierarchy is that originally /sbin binaries were kept in /etc.

```
Deciding what things go into "sbin" directories is simple: if a normal
(not a system administrator) user will ever run it directly, then it
must be placed in one of the "bin" directories. Ordinary users should
not have to place any of the sbin directories in their path.
```

```
For example, files such as chfn which users only occasionally use must
still be placed in /usr/bin. ping, although it is absolutely necessary
for root (network recovery and diagnosis) is often used by users and
must live in /bin for that reason.
```

```
We recommend that users have read and execute permission for everything
in /sbin except, perhaps, certain setuid and setgid programs. The
division between /bin and /sbin was not created for security reasons or
to prevent users from seeing the operating system, but to provide a
good partition between binaries that everyone uses and ones that are
primarily used for administration tasks. There is no inherent security
advantage in making /sbin off-limits for users.
```

FSSTND compliance requires that the following commands, or symbolic links to commands, are required in /sbin.

```
shutdown Command to bring the system down.
```

The following files, or symbolic links to files, must be in /sbin if the corresponding subsystem is installed:

fastboot	Reboot the system without checking the disks (optional)
fasthalt	Stop the system without checking the disks (optional)
fdisk	Partition table manipulator (optional)
fsck	File system check and repair utility (optional)

Linux Filesystem Hierarchy

fsck.*	File system check and repair utility for a specific filesystem (optional)
getty	The getty program (optional)
halt	Command to stop the system (optional)
ifconfig	Configure a network interface (optional)
init	Initial process (optional)
mkfs	Command to build a filesystem (optional)
mkfs.*	Command to build a specific filesystem (optional)
mkswap	Command to set up a swap area (optional)
reboot	Command to reboot the system (optional)
route	IP routing table utility (optional)
swapon	Enable paging and swapping (optional)
swapoff	Disable paging and swapping (optional)
update	Daemon to periodically flush filesystem buffers (optional)

1.17. /usr

/usr usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.... X and its supporting libraries can be found here. User programs like telnet, ftp, etc.... are also placed here. In the original Unix implementations, /usr was where the home directories of the users were placed (that is to say, /usr/someone was then the directory now known as /home/someone). In current Unices, /usr is where user-land programs and data (as opposed to 'system land' programs and data) are. The name hasn't changed, but it's meaning has narrowed and lengthened from "everything user related" to "user usable programs and data". As such, some people may now refer to this directory as meaning 'User System Resources' and not 'user' as was originally intended.

/usr is shareable, read-only data. That means that /usr should be shareable between various FHS-compliant hosts and must not be written to. Any information that is host-specific or varies with time is stored elsewhere.

Large software packages must not use a direct subdirectory under the /usr hierarchy.

/usr/X11R6

Another large subdirectory structure begins here, containing libraries, executables, docs, fonts and much more concerning the X Window System. Its inclusion here is somewhat inconsistent and so is the difference between '/usr' and '/usr/X11R6' directories. One would assume that programs that run on X only have their files in the '/usr/X11R6' hierarchy, while the others use '/usr'. Regrettably, it isn't so. KDE and GNOME put their files in the '/usr' hierarchy, whereas the window manager Window Maker uses '/usr/X11R6'. Documentation files for X11R6 are not in '/usr/X11R6/doc', but primarily in '/usr/X11R6/lib/X11/doc'. This mess is due to the fact that in contrast to other operating systems, the graphical desktop isn't an integral part of the system. Linux is still primarily used on servers, where graphical systems don't make sense.

This hierarchy is reserved for the X Window System, version 11 release 6, and related files. To simplify matters and make XFree86 more compatible with the X Window System on other systems, the following symbolic links must be present if /usr/X11R6 exists:

```
/usr/bin/X11 -> /usr/X11R6/bin  
/usr/lib/X11 -> /usr/X11R6/lib/X11  
/usr/include/X11 -> /usr/X11R6/include/X11
```

In general, software must not be installed or managed via the above symbolic links. They are intended for utilization by users only. The difficulty is related to the release version of the X

Linux Filesystem Hierarchy

Window System – in transitional periods, it is impossible to know what release of X11 is in use.

/usr/X11R6/bin

XFree86 system binaries. These are necessary for the initialisation, configuration and running of the X windowing system. X, xf86config, xauth, xmodmap and even xpenguin are located here.

/usr/X11R6/include

XFree86 system header files. There are required for the compilation of some applications that utilise the X toolkit.

/usr/X11R6/lib

XFree86 system libraries.

/usr/X11R6/lib/modules

XFree86 system modules. These are the modules that X loads upon startup. Without these modules video4linux, DRI and GLX extensions and drivers for certain input devices would cease to function.

/usr/X11R6/lib/X11/fonts

XFree86 system fonts. Fonts that are utilised by 'xfs' (the X Font Server) and programs of that ilk.

/usr/bin

This directory contains the vast majority of binaries on your system. Executables in this directory vary widely. For instance vi, gcc, gnome-session and mozilla and are all found here.

/usr/doc

The central documentation directory. Documentation is actually located in /usr/share/doc and linked from here.

/usr/etc

Theoretically, that's another directory for configuration files. Virtually unused now.

/usr/games

Once upon a time, this directory contained network games files. Rarely used now.

/usr/include

The directory for 'header files', needed for compiling user space source code.

/usr/include/'package-name'

Application specific header files.

/usr/info

This directory used to contain the files for the info documentation system. Now they are in '/usr/share/info'.

/usr/lib

This directory contains program libraries. Libraries are collections of frequently used program routines.

/usr/local

The original idea behind '/usr/local' was to have a separate ('local') '/usr' directory on every machine besides '/usr', which might be just mounted read-only from somewhere else. It copies the structure of '/usr'. These days, '/usr/local' is widely regarded as a good place in which to keep self-compiled or third-party programs. The /usr/local hierarchy is for use by the system administrator when installing software locally. It needs to be safe from being overwritten when the system software is updated. It may be used for programs and data that are shareable amongst a group of hosts, but not found in /usr. Locally installed software must be placed within /usr/local rather than /usr unless it is being installed to replace or upgrade software in /usr.

/usr/man

It once held the man pages. It has been moved to /usr/share/man.

/usr/sbin

This directory contains programs for administering a system, meant to be run by 'root'. Like '/sbin', it's not part of a user's \$PATH. Examples of included binaries here are chroot, useradd, in.tftpd and pppconfig.

/usr/share

Linux Filesystem Hierarchy

This directory contains 'shareable', architecture-independent files (docs, icons, fonts etc). Note, however, that '/usr/share' is generally not intended to be shared by different operating systems or by different releases of the same operating system. Any program or package which contains or requires data that doesn't need to be modified should store that data in '/usr/share' (or '/usr/local/share', if installed locally). It is recommended that a subdirectory be used in /usr/share for this purpose."

/usr/share/doc

Location of package specific documentation files. These directories often contain useful information that might not be in the man pages. They may also contain templates and configuration files for certain utilities making configuration that much easier.

/usr/share/info

Location of 'info' pages. This style of documentation seems to be largely ignored now. Manual pages are in far greater favour.

/usr/share/man

Manual pages. They are organised into 8 sections, which are explained below.

man1: User programs

Manual pages that describe publicly accessible commands are contained in this chapter. Most program documentation that a user will need to use is located here.

man2: System calls

This section describes all of the system calls (requests for the kernel to perform operations).

man3: Library functions and subroutines

Section 3 describes program library routines that are not direct calls to kernel services. This and chapter 2 are only really of interest to programmers.

man4: Special files

Section 4 describes the special files, related driver functions, and networking support available in the system. Typically, this includes the device files found in /dev and the kernel interface to networking protocol support.

man5: File formats

The formats for many data files are documented in the section 5. This includes various include files, program output files, and system files.

man6: Games

This chapter documents games, demos, and generally trivial programs. Different people have various notions about how essential this is.

man7: Miscellaneous

Manual pages that are difficult to classify are designated as being section 7. The troff and other text processing macro packages are found here.

man8: System administration Programs

used by system administrators for system operation and maintenance are documented here. Some of these programs are also occasionally useful for normal users.

/usr/src

The 'linux' sub-directory holds the Linux kernel sources, header-files and documentation.

/usr/src/RPM

RPM provides a substructure for building RPMs from SRPMs. Organisation of this branch is fairly logical with packages being organised according to a package's architecture.

/usr/src/RPM/BUILD

A temporary store for RPM binary files that are being built from source code.

Linux Filesystem Hierarchy

/usr/src/RPM/RPMS/athlon, /usr/src/RPM/RPMS/i386, /usr/src/RPM/RPMS/i486, /usr/src/RPM/RPMS/i586,
/usr/src/RPM/RPMS/i686, /usr/src/RPM/RPMS/noarch

These directories contain architecture dependant RPM source files.

/usr/src/RPM/SOURCES

This directory contains the source TAR files, patches, and icon files for software to be packaged.

/usr/src/RPM/SPECS

RPM SPEC files. A SPEC file is a file that contains information as well as scripts that are necessary to build a package.

/usr/src/RPM/SRPMS

Contains the source RPM files which result from a build.

/usr/src/linux

Contains the source code for the Linux kernel.

/usr/src/linux/.config

The last kernel source configuration. This file is normally created through the 'make config', 'make menuconfig' or 'make xconfig' steps during kernel compilation.

/usr/src/linux/.depend, /usr/src/linux/.hdepend

'make dep' checks the dependencies of the selections you made when you created your .config file. It ensures that the required files can be found and it builds a list that is to be used during compilation.

Should this process be successful these two files are created.

/usr/src/linux/COPYING

GNU License

/usr/src/linux/CREDITS

A partial credits–file of people that have contributed to the Linux project. It is sorted by name and formatted to allow easy grepping and beautification by scripts. The fields are: name (N), email (E), web–address (W), PGP key ID and fingerprint (P), description (D), and snail–mail address (S).

/usr/src/linux/MAINTAINERS

List of maintainers and details on how to submit kernel changes.

/usr/src/linux/Makefile

Contains data necessary for compilation of a working kernel. It allows developers and end–users to compile a kernel with a few simple steps (ie. make dep, make clean, make bzImage, make modules, make modules_install) and also not have to worry about re–compiling everything from scratch if parts of it have already been done so and are up to date.

/usr/src/linux/README

These are the release notes for Linux version 2.4. Read them carefully, as they tell you what this is all about, explain how to install the kernel, and what to do if something goes wrong.

/usr/src/linux/REPORTING–BUGS

A suggested procedure for reporting Linux bugs. You aren't obliged to use the bug reporting format, it is provided as a guide to the kind of information that can be useful to developers – no more.

/usr/src/linux/Rules.make

This file contains rules which are shared between multiple Makefiles.

/usr/src/linux/Documentation

Contains documentation that may be necessary in order to re–compile a kernel. However, it also provides quite a lot of information about your Linux system in general as well. For those who wish to seek further information on the contents of this directory you may consult the

/usr/src/linux/Documentation/00–INDEX file. Further, more detailed documentation may be found in /usr/src/linux/Documentation/Docbook. Of course, the contents of this directory is written in Docbook but may be converted to pdf, ps or html using the make targets of 'pdfdocs', 'psdocs' and 'htmldocs' respectively.

/usr/tmp

User space temporary files. This directory is not found on modern distributions at all and was most likely created as a consequence of Linux's UNIX heritage.

1.18. /var

Contains variable data like system logging files, mail and printer spool directories, and transient and temporary files. Some portions of /var are not shareable between different systems. For instance, /var/log, /var/lock, and /var/run. Other portions may be shared, notably /var/mail, /var/cache/man, /var/cache/fonts, and /var/spool/news. Why not put it into /usr? Because there might be circumstances when you may want to mount /usr as read-only, e.g. if it is on a CD or on another computer. '/var' contains variable data, i.e. files and directories the system must be able to write to during operation, whereas /usr should only contain static data. Some directories can be put onto separate partitions or systems, e.g. for easier backups, due to network topology or security concerns. Other directories have to be on the root partition, because they are vital for the boot process. 'Mountable' directories are: '/home', '/mnt', '/tmp', '/usr' and '/var'. Essential for booting are: '/bin', '/boot', '/dev', '/etc', '/lib', '/proc' and '/sbin'.

If /var cannot be made a separate partition, it is often preferable to move /var out of the root partition and into the /usr partition. (This is sometimes done to reduce the size of the root partition or when space runs low in the root partition.) However, /var must not be linked to /usr because this makes separation of /usr and /var more difficult and is likely to create a naming conflict. Instead, link /var to /usr/var.

Applications must generally not add directories to the top level of /var. Such directories should only be added if they have some system-wide implication, and in consultation with the FHS mailing list.

/var/backups

Directory containing backups of various key system files such as /etc/shadow, /etc/group, /etc/inetd.conf and dpkg.status. They are normally renamed to something like dpkg.status.0, group.bak, gshadow.bak, inetd.conf.bak, passwd.bak, shadow.bak

/var/cache

Is intended for cached data from applications. Such data is locally generated as a result of time-consuming I/O or calculation. This data can generally be regenerated or be restored. Unlike /var/spool, files here can be deleted without data loss. This data remains valid between invocations of the application and rebooting of the system. The existence of a separate directory for cached data allows system administrators to set different disk and backup policies from other directories in /var.

/var/cache/fonts

Locally-generated fonts. In particular, all of the fonts which are automatically generated by mktexpk must be located in appropriately-named subdirectories of /var/cache/ fonts.

/var/cache/man

A cache for man pages that are formatted on demand. The source for manual pages is usually stored in /usr/share/man/; some manual pages might come with a pre-formatted version, which is stored in /usr/share/man/cat* (this is fairly rare now). Other manual pages need to be formatted when they are first viewed; the formatted version is then stored in /var/man so that the next person to view the same page won't have to wait for it to be formatted (/var/catman is often cleaned in the same way temporary directories are cleaned).

/var/cache/'package-name'

Package specific cache data.

/var/cache/www

WWW proxy or cache data.

/var/crash

Linux Filesystem Hierarchy

This directory will eventually hold system crash dumps. Currently, system crash dumps are not supported under Linux. However, development is already complete and is awaiting unification into the Linux kernel.

/var/db

Data bank store.

/var/games

Any variable data relating to games in /usr is placed here. It holds variable data that was previously found in /usr. Static data, such as help text, level descriptions, and so on, remains elsewhere though, such as in /usr/share/games. The separation of /var/games and /var/lib as in release FSSTND 1.2 allows local control of backup strategies, permissions, and disk usage, as well as allowing inter-host sharing and reducing clutter in /var/lib. Additionally, /var/games is the path traditionally used by BSD.

/var/lib

Holds dynamic data libraries/files like the rpm/dpkg database and game scores. Furthermore, this hierarchy holds state information pertaining to an application or the system. State information is data that programs modify while they run, and that pertains to one specific host. Users shouldn't ever need to modify files in /var/lib to configure a package's operation. State information is generally used to preserve the condition of an application (or a group of inter-related applications) between invocations and between different instances of the same application. An application (or a group of inter-related applications) use a subdirectory of /var/lib for their data. There is one subdirectory, /var/lib/misc, which is intended for state files that don't need a subdirectory; the other subdirectories should only be present if the application in question is included in the distribution. /var/lib/'name' is the location that must be used for all distribution packaging support. Different distributions may use different names, of course.

/var/local

Variable data for local programs (i.e., programs that have been installed by the system administrator) that are installed in /usr/local (as opposed to a remotely mounted '/var' partition). Note that even locally installed programs should use the other /var directories if they are appropriate, e.g., /var/lock.

/var/lock

Many programs follow a convention to create a lock file in /var/lock to indicate that they are using a particular device or file. This directory holds those lock files (for some devices) and hopefully other programs will notice the lock file and won't attempt to use the device or file.

Lock files should be stored within the /var/lock directory structure. Lock files for devices and other resources shared by multiple applications, such as the serial device lock files that were originally found in either /usr/spool/locks or /usr/spool/uucp, must now be stored in /var/lock. The naming convention which must be used is LCK.. followed by the base name of the device file. For example, to lock /dev/ttyS0 the file LCK..ttyS0 would be created. The format used for the contents of such lock files must be the HDB UUCP lock file format. The HDB format is to store the process identifier (PID) as a ten byte ASCII decimal number, with a trailing newline. For example, if process 1230 holds a lock file, it would contain the eleven characters: space, space, space, space, space, space, one, two, three, zero, and newline.

/var/log

Log files from the system and various programs/services, especially login (/var/log/wtmp, which logs all logins and logouts into the system) and syslog (/var/log/messages, where all kernel and system program message are usually stored). Files in /var/log can often grow indefinitely, and may require cleaning at regular intervals. Something that is now normally managed via log rotation utilities such as 'logrotate'. This utility also allows for the automatic rotation compression, removal and mailing of log files. Logrotate can be set to handle a log file daily, weekly, monthly or when the log file gets to a certain size. Normally, logrotate runs as a daily cron job. This is a good place to start troubleshooting general technical problems.

Linux Filesystem Hierarchy

/var/log/auth.log	Record of all logins and logouts by normal users and system processes.
/var/log/btmp	Log of all attempted bad logins to the system. Accessed via the lastb command.
/var/log/debug	Debugging output from various packages.
/var/log/dmesg	Kernel ring buffer. The content of this file is referred to by the dmesg command.
/var/log/gdm/	GDM log files. Normally a subset of the last X log file. See /var/log/xdm.log for mode details.
/var/log/kdm.log	KDM log file. Normally a subset of the last X log file. See /var/log/xdm.log for more details.
/var/log/messages	System logs.
/var/log/pacct	Process accounting is the bookkeeping of process activity. The raw data of process activity is maintained here. Three commands can be used to access the contents of this file dump-acct, sa (summary of process accounting) and lastcomm (list the commands executed on the system).
/var/log/utmp	Active user sessions. This is a data file and as such it can not be viewed normally. A human-readable form can be created via the dump-utmp command or through the w, who or users commands.
/var/log/wtmp	Log of all users who have logged into and out of the system. The last command can be used to access a human readable form of this file. It also lists every connection and run-level change.
/var/log/xdm.log	XDM log file. Normally subset of the last X startup log and pretty much useless in light of the details the X logs is able to provide us with. Only consult this file if you have XDM specific issues otherwise just use the X logfile.
/var/log/XFree86.0.log, /var/log/XFree86.?.log	X startup logfile. An excellent resource for uncovering problems with X configuration. Log files are numbered according to when they were last used. For example, the last log file would be stored in /var/log/XFree86.0.log, the next /var/log/XFree86.9.log, so on and so forth.
/var/log/syslog	The 'system' log file. The contents of this file is managed via the syslogd daemon which more often than not takes care of all log manipulation on most systems.
/var/mail	Contains user mailbox files. The mail files take the form /var/mail/'username' (Note that /var/mail may be a symbolic link to another directory). User mailbox files in this location are stored in the standard UNIX mailbox format. The reason for the location of this directory was to bring the FHS inline with nearly every UNIX implementation (it was previously located in /var/spool/mail). This change is important for inter-operability since a single /var/mail is often shared between multiple hosts and multiple UNIX implementations (despite NFS locking issues).
/var/opt	Variable data of the program packages in /opt must be installed in /var/opt/'package-name', where 'package-name' is the name of the subtree in /opt where the static data from an add-on software package is stored, except where superceded by another file in /etc. No structure is imposed on the internal arrangement of /var/opt/'package-name'.
/var/run	Contains the process identification files (PIDs) of system services and other information about the system that is valid until the system is next booted. For example, /var/run/utmp contains information about users currently logged in.

Linux Filesystem Hierarchy

/var/spool

Holds spool files, for instance for mail, news, and printing (lpd) and other queued work. Spool files store data to be processed after the job currently occupying a device is finished or the appropriate cron job is started. Each different spool has its own subdirectory below /var/spool, e.g., the cron tables are stored in /var/spool/cron/crontabs.

/var/tmp

Temporary files that are large or that need to exist for a longer time than what is allowed for /tmp.
(Although the system administrator might not allow very old files in /var/tmp either.)

/var/named

Database for BIND. The Berkeley Internet Name Domain (BIND) implements an Internet domain name server. BIND is the most widely used name server software on the Internet, and is supported by the Internet Software Consortium, www.isc.org.

/var/yp

Database for NIS (Network Information Services). NIS is mostly used to let several machines in a network share the same account information (eg. /etc/passwd). NIS was formerly called Yellow Pages (YP).

The following directories, or symbolic links to directories, are required in /var for FSSTND compliance:

/var/cache	Application cache data
/var/lib	Variable state information
/var/local	Variable data for /usr/local
/var/lock	Lock files
/var/log	Log files and directories
/var/opt	Variable data for /opt
/var/run	Data relevant to running processes
/var/spool	Application spool data
/var/tmp	Temporary files preserved between system reboots

Several directories are 'reserved' in the sense that they must not be used arbitrarily by some new application, since they would conflict with historical and/or local practice. They are:

/var/backups
/var/cron
/var-msgs
/var/preserve

The following directories, or symbolic links to directories, must be in /var, if the corresponding subsystem is installed:

account	Process accounting logs (optional)
crash	System crash dumps (optional)
games	Variable game data (optional)
mail	User mailbox files (optional)
yp	Network Information Service (NIS) database files (optional)

1.19. /srv

/srv contains site-specific data which is served by this system.

This main purpose of specifying this is so that users may find the location of the data files for particular service, and so that services which require a single tree for readonly data, writable data

Linux Filesystem Hierarchy

and scripts (such as cgi scripts) can be reasonably placed. Data that is only of interest to a specific user should go in that users' home directory.

The methodology used to name subdirectories of /srv is unspecified as there is currently no consensus on how this should be done. One method for structuring data under /srv is by protocol, eg. ftp, rsync, www, and cvs. On large systems it can be useful to structure /srv by administrative context, such as /srv/physics/www, /srv/compsci/cvs, etc. This setup will differ from host to host. Therefore, no program should rely on a specific subdirectory structure of /srv existing or data necessarily being stored in /srv. However /srv should always exist on FHS compliant systems and should be used as the default location for such data.

Distributions must take care not to remove locally placed files in these directories without administrator permission.

This is particularly important as these areas will often contain both files initially installed by the distributor, and those added by the administrator.

1.20. /tmp

This directory contains mostly files that are required temporarily. Many programs use this to create lock files and for temporary storage of data. Do not remove files from this directory unless you know exactly what you are doing! Many of these files are important for currently running programs and deleting them may result in a system crash. Usually it won't contain more than a few KB anyway. On most systems, this directory is cleared out at boot or at shutdown by the local system. The basis for this was historical precedent and common practice. However, it was not made a requirement because system administration is not within the scope of the FSSTND. For this reason people and programs must not assume that any files or directories in /tmp are preserved between invocations of the program. The reasoning behind this is for compliance with IEEE standard P1003.2 (POSIX, part 2).

Glossary

ARPA

The Advanced Research and Projects Agency of the United States Department of Defense. Also known as DARPA (the "D" stands for "Defense"), it originated in the late 1960s and early 1970s the proposal and standards for the Internet. For this reason, the Internet was initially referred to as ARPANet, and connected the military with the various centers of research around the United States in a way that was intended to have a high degree of survivability against a nuclear attack.

BASH

The Bourne Again Shell and is based on the Bourne shell, sh, the original command interpreter.

Bourne Shell

The Bourne shell is the original Unix shell (command execution program, often called a command interpreter) that was developed at AT&T. Named for its developer, Stephen Bourne, the Bourne shell is also known by its program name, sh. The shell prompt (character displayed to indicate readiness for input) used is the \$ symbol. The Bourne shell family includes the Bourne, Korn shell, bash, and zsh shells. Bourne Again Shell (bash) is the free version of the Bourne shell distributed with Linux systems. Bash is similar to the original, but has added features such as command line editing. Its name is sometimes spelled as Bourne Again SHell, the capitalized Hell referring to the difficulty some people have with it.

CLI

A CLI (command line interface) is a user interface to a computer's operating system or an application in which the user responds to a visual prompt by typing in a command on a specified line, receives a response back from the system, and then enters another command, and so forth. The MS-DOS Prompt application in a Windows operating system is an example of the provision of a command line interface. Today, most users prefer the graphical user interface (GUI) offered by Windows, Mac OS, BeOS, and others. Typically, most of today's Unix-based systems offer both a command line interface and a graphical user interface.

core

A core file is created when a program terminates unexpectedly, due to a bug, or a violation of the operating system's or hardware's protection mechanisms. The operating system kills the program and creates a core file that programmers can use to figure out what went wrong. It contains a detailed description of the state that the program was in when it died. If you would like to determine what program a core file came from, use the file command, like this: \$ file core That will tell you the name of the program that produced the core dump. You may want to write the maintainer(s) of the program, telling them that their program dumped core. To Enable or Disable Core Dumps you must use the ulimit command in bash, the limit command in tcsh, or the rlimit command in ksh. See the appropriate manual page for details. This setting affects all programs run from the shell (directly or indirectly), not the whole system. If you wish to enable or disable core dumping for all processes by default, you can change the default setting in /usr/include/linux/sched.h. Refer to definition of INIT_TASK, and look also in /usr/include/linux/resource.h. PAM support optimizes the system's environment, including the amount of memory a user is allowed. In some distributions this parameter is configurable in the /etc/security/limits.conf file. For more information, refer to the Linux Administrator's Security Guide.

daemon

A process lurking in the background, usually unnoticed, until something triggers it into action. For example, the \cmd{update} daemon wakes up every thirty seconds or so to flush the buffer cache, and the \cmd{sendmail} daemon awakes whenever someone sends mail.

DARPA

The Defense Advanced Research Projects Agency is the central research and development organization for the Department of Defense (DoD). It manages and directs selected basic and applied

Linux Filesystem Hierarchy

research and development projects for DoD, and pursues research and technology where risk and payoff are both very high and where success may provide dramatic advances for traditional military roles and missions.

DHCP

Dynamic Host Control Protocol, is a protocol like BOOTP (actually dhcpcd includes much of the functionality of BOOTPD). It assigns IP addresses to clients based on lease times. DHCP is used extensively by Microsoft and more recently also by Apple. It is probably essential in any multi-platform environment.

DNS

Domain Name System translates Internet domain and host names to IP addresses. DNS implements a distributed database to store name and address information for all public hosts on the Net. DNS assumes IP addresses do not change (i.e., are statically assigned rather than dynamically assigned). The DNS database resides on a hierarchy of special-purpose servers. When visiting a Web site or other device on the Net, a piece of software called the DNS resolver (usually built into the network operating system) first contacts a DNS server to determine the server's IP address. If the DNS server does not contain the needed mapping, it will in turn forward the request to a DNS server at the next higher level in the hierarchy. After potentially several forwarding and delegation messages are sent within the DNS hierarchy, the IP address for the given host eventually is delivered to the resolver. DNS also includes support for caching requests and for redundancy. Most network operating systems allow one to enter the IP addresses of primary, secondary, and tertiary DNS servers, each of which can service initial requests from clients. Many ISPs maintain their own DNS servers and use DHCP to automatically assign the addresses of these servers to dial-in clients, so most home users need not be aware of the details behind DNS configuration. Registered domain names and addresses must be renewed periodically, and should a dispute occur between two parties over ownership of a given name, such as in trademarking, ICANN's Uniform Domain-Name Dispute-Resolution Policy can be invoked. Also known as Domain Name System, Domain Name Service, Domain Name Server.

environment variable

A variable that is available to any program that is started by the shell.

ESD

Enlightened Sound Daemon. This program is designed to mix together several digitized audio streams for playback by a single device.

filesystem

The methods and data structures that an operating system uses to keep track of files on a disk or partition; the way the files are organized on the disk. Also used to describe a partition or disk that is used to store the files or the type of the filesystem.

FSSTND

Often the group, which creates the Linux File System Structure document, or the document itself, is referred to as the 'FSSTND'. This is short for "file system standard". This document has helped to standardize the layout of file systems on Linux systems everywhere. Since the original release of the standard, most distributors have adopted it in whole or in part, much to the benefit of all Linux users. It is now often referred to as the FHS (Filesystem Hierarchy Standard) document though since its incorporation into the LSB (Linux Standards Base) Project.

GUI

Graphical User Interface. The use of pictures rather than just words to represent the input and output of a program. A program with a GUI runs under some windowing system (e.g. The X Window System, Microsoft Windows, Acorn RISC OS, NEXTSTEP). The program displays certain icons, buttons, dialogue boxes etc. in its windows on the screen and the user controls it mainly by moving a pointer on the screen (typically controlled by a mouse) and selecting certain objects by pressing buttons on the mouse while the pointer is pointing at them. Though Apple Computer would like to claim they invented the GUI with their Macintosh operating system, the concept originated in the early 1970s at Xerox's PARC laboratory.

Linux Filesystem Hierarchy

hard link

A directory entry, which maps a filename to an inode number. A file may have multiple names or hard links. The link count gives the number of names by which a file is accessible. Hard links do not allow multiple names for directories and do not allow multiple names in different filesystems.

init

'init' process is the first user level process started by the kernel. init has many important duties, such as starting getty (so that users can log in), implementing run levels, and taking care of orphaned processes. This chapter explains how init is configured and how you can make use of the different run levels. init is one of those programs that are absolutely essential to the operation of a Linux system, but that you still can mostly ignore. Usually, you only need to worry about init if you hook up serial terminals, dial-in (not dial-out) modems, or if you want to change the default run level. When the kernel has started (has been loaded into memory, has started running, and has initialized all device drivers and data structures and such), it finishes its own part of the boot process by starting a user level program, init. Thus, init is always the first process (its process number is always 1). The kernel looks for init in a few locations that have been historically used for it, but the proper location for it is /sbin/init. If the kernel can't find init, it tries to run /bin/sh, and if that also fails, the startup of the system fails. When init starts, it completes the boot process by doing a number of administrative tasks, such as checking filesystems, cleaning up /tmp, starting various services, and starting a getty for each terminal and virtual console where users should be able to log in. After the system is properly up, init restarts getty for each terminal after a user has logged out (so that the next user can log in). init also adopts orphan processes: when a process starts a child process and dies before its child, the child immediately becomes a child of init. This is important for various technical reasons, but it is good to know it, since it makes it easier to understand process lists and process tree graphs. init itself is not allowed to die. You can't kill init even with SIGKILL. There are a few variants of init available. Most Linux distributions use sysvinit (written by Miquel van Smoorenburg), which is based on the System V init design. The BSD versions of Unix have a different init. The primary difference is run levels: System V has them, BSD doesn't.

inode

An inode is the address of a disk block. When you see the inode information through ls, ls prints the address of the first block in the file. You can use this information to tell if two files are really the same file with different names (links). A file has several components: a name, contents, and administrative information such as permissions and modification times. The administrative information is stored in the inode (over the years, the hyphen fell out of "i-node"), along with essential system data such as how long it is, where on the disc the contents of the file are stored, and so on. There are three times in the inode: the time that the contents of the file were last modified (written); the time that the file was last used (read or executed); and the time that the inode itself was last changed, for example to set the permissions. Altering the contents of the file does not affect its usage time and changing the permissions affects only the inode change time. It is important to understand inodes, not only to appreciate the options on ls, but because in a strong sense the inodes are the files. All the directory hierarchy does is provide convenient names for files. The system's internal name for the file is its i-number: the number of the inode holding the file's information.

kernel

Part of an operating system that implements the interaction with hardware and the sharing of resources.

libraries

Executables should have no undefined symbols, only useful symbols; all useful programs refer to symbols they do not define (eg. printf or write). These references are resolved by pulling object files from libraries into the executable.

link

A symbolic link (alias in MacOS and shortcut under Windows) is a file that points to another file; this is a commonly used tool. A hard-link rarely created by the user, is a filename that points to a block of

Linux Filesystem Hierarchy

data that has several other filenames as well.

man page

Every version of UNIX comes with an extensive collection of online help pages called man pages (short for manual pages). The man pages are the authoritative documentation about your UNIX system. They contain complete information about both the kernel and all the utilities.

MTA

Mail Transfer Agents. Alongside the web, mail is the top reason for the popularity of the Internet. E-mail is an inexpensive and fast method of time-shifted messaging which, much like the Web, is actually based around sending and receiving plain text files. The protocol used is called the Simple Mail Transfer Protocol (SMTP). The server programs that implement SMTP to move mail from one server to another are called MTAs. Once upon a time users would have to Telnet into an SMTP server and use a command line mail program like 'mutt' or 'pine' to check their mail. Now, GUI based e-mail clients like Mozilla, Kmail and Outlook allow users to check their email off of a local SMTP sever. Additional protocols like POP3 and IMAP4 are used between the SMTP server and desktop mail client to allow clients to manipulate files on, and download from, their local mail server. The programs that implement POP3 and IMAP4 are called Mail Delivery Agents (MDAs). They are generally separate from MTAs.

NFS

Network File System, is the UNIX equivalent of Server Message Block (SMB). It is a way through which different machines can import and export local files between each other. Like SMB though, NFS sends information including user passwords unencrypted, so it's best to limit its usage to within your local network.

operating system

Software that shares a computer system's resources (processor, memory, disk space, network bandwidth, and so on) between users and the application programs they run. Controls access to the system to provide security.

PAM

Pluggable Authentication Modules. A suite of shared libraries that determine how a user will be authenticated. For example, conventionally UNIX users authenticate themselves by supplying a password at the password prompt after they have typed their name at the login prompt. In many circumstances, such as internal access to workstations, this simple form of authentication is considered sufficient. In other cases, more information is warranted. If a user wants to log in to an internal system from an external source, like the Internet, more or alternative information may be required – perhaps a one-time password. PAM provides this type of capability and much more. Most important, PAM modules allow you to configure your environment with the necessary level of security.

PATH

The shell looks for commands and programs in a list of file paths stored in the PATH environment variable. An environment variable stores information in a place where other programs and commands can access it. Environment variables store information such as the shell that you are using, your login name, and your current working directory. To see a list of all the environment variables currently defined; type 'set' at the prompt. When you type a command at the shell prompt, the shell will look for that command's program file in each directory listed in the PATH variable, in order. The first program found matching the command you typed will be run. If the command's program file is not in a directory listed in your PATH environment variable, the shell returns a "commands not found" error. By default, the shell does not look in your current working directory or your home directory for commands. This is really a security mechanism so that you don't execute programs by accident. What if a malicious user put a harmful program called ls in your home directory? If you typed ls and the shell looked for the fake program in your home directory before the real program in the /bin directory, what do you think would happen? If you thought bad things, you are on the right track. Since your PATH doesn't have the current directory as one of its search locations, programs in your current

Linux Filesystem Hierarchy

directory must be called with an absolute path of a relative path specified as './program-name'. To see what directories are part of your PATH enter this command: # echo \$PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11

pipes and sockets

Special files that programs use to communicate with one another. They are rarely seen, but you might be able to see a socket or two in the /dev/ directory.

process identifier

Shown in the heading of the ps command as PID. The unique number assigned to every process running in the system.

rpc

Remote Procedure Calls. It enables a system to make calls to programs such as NFS across the network transparently, enabling each system to interpret the calls as if they were local. In this case, it would make exported filesystems appear as though they were local.

set group ID (SGID)

The SGID permission causes a script to run with its group set to the group of the script, rather than the group of the user who started it. It is normally considered extremely bad practice to run a program in this way as it can pose many security problems. Later versions of the Linux kernel will even prohibit the running of shell scripts that have this attribute set.

set user ID (SUID)

The SUID permission causes a script to run as the user who is the owner of the script, rather than the user who started it. It is normally considered extremely bad practice to run a program in this way as it can pose many security problems. Later versions of the Linux kernel will even prohibit the running of shell scripts that have this attribute set.

signal

Software interrupts sent to a program to indicate that an important event has occurred. The events can vary from user requests to illegal memory access errors. Some signals, like the interrupt signal, indicate that a user has asked the program to do something that is not in the usual flow of control.

SSH

The Secure Shell, or SSH, provides a way of running command line and graphical applications, and transferring files, over an encrypted connection, all that will be seen is junk. It is both a protocol and a suite of small command line applications, which can be used for various functions. SSH replaces the old Telnet application, and can be used for secure remote administration of machines across the Internet. However, it also has other features. SSH increases the ease of running applications remotely by setting up X permissions automatically. If you can log into a machine, it allows you to run a graphical application on it, unlike Telnet, which requires users to have an understanding of the X authentication mechanisms that are manipulated through the xauth and xhost commands. SSH also has inbuilt compression, which allows your graphic applications to run much faster over the network. SCP (Secure Copy) and SFTP (Secure FTP) allow transfer of files over the remote link, either via SSH's own command line utilities or graphical tools like Gnome's GFTP. Like Telnet, SSH is cross-platform. You can find SSH server and clients for Linux, Unix and all flavours of Windows, BeOS, PalmOS, Java and embedded Oses used in routers.

STDERR

Standard error. A special type of output used for error messages. The file descriptor for STDERR is 2.

STDIN

Standard input. User input is read from STDIN. The file descriptor for STDIN is 0.

STDOUT

Standard output. The output of scripts is usually to STDOUT. The file descriptor for STDOUT is 1.

symbol table

The part of an object table that gives the value of each symbol (usually as a section name and an offset) is called the symbol table. Executables may also have a symbol table, with this one giving the final values of the symbols. Debuggers use the symbol table to present addresses to the user in a

Linux Filesystem Hierarchy

symbolic, rather than a numeric form. It is possible to strip the symbol table from executables resulting in a smaller sized executable but this prevents meaningful debugging.

symbolic link or soft link

A special filetype, which is a small pointer file, allowing multiple names for the same file. Unlike hard links, symbolic links can be made for directories and can be made across filesystems. Commands that access the file being pointed to are said to follow the symbolic link. Commands that access the link itself do not follow the symbolic link.

system call

The services provided by the kernel to application programs, and the way in which they are invoked. See section 2 of the manual pages.

system program

Programs that implement high level functionality of an operating system, i.e., things that aren't directly dependent on the hardware. May sometimes require special privileges to run (e.g., for delivering electronic mail), but often just commonly thought of as part of the system (e.g., a compiler).

tcp-wrappers

Almost all of the services provided through inetd are invoked through tcp-wrappers by way of the tcp-wrappers daemon, tcpd. The tcp-wrappers mechanism provides access control list restrictions and logging for all service requests to the service it wraps. It may be used for either TCP or TCP services as long as the services are invoked through a central daemon process such as inetd. These programs log the client host name of incoming telnet, ftp, rsh, rlogin, finger etc.... requests. Security options are access control per host, domain and/or service; detection of host name spoofing or host address spoofing; booby traps to implement an early-warning system.

ZSH

Zsh was developed by Paul Falstad as a replacement for both the Bourne and C shell. It incorporates features of all the other shells (such as file name completion and a history mechanism) as well as new capabilities. Zsh is considered similar to the Korn shell. Falstad intended to create in zsh a shell that would do whatever a programmer might reasonably hope it would do. Zsh is popular with advanced users. Along with the Korn shell and the C shell, the Bourne shell remains among the three most widely used and is included with all UNIX systems. The Bourne shell is often considered the best shell for developing scripts.

Appendix A. UNIX System V Signals

<i>Symbol</i>	<i>Number</i>	<i>Action</i>	<i>Meaning</i>
SIGHUP	1	exit	Hangs up.
SIGINT	2	exit	Interrupts.
SIGQUIT	3	core dump	Quits.
SIGILL	4	core dump	Illegal instruction.
SIGTRAP	5	core dump	Trace trap.
SIGIOT	6	core dump	IOT instruction.
SIGEMT	7	core dump	MT instruction.
SIGFPE	8	core dump	Floating point exception.
SIGKILL	9	exit	Kills (cannot be caught or ignored).
SIGBUS	10	core dump	Bus error.
SIGSEGV	11	core dump	Segmentation violation.
SIGSYS	12	core dump	Bad argument to system call.
SIGPIPE	13	exit	Writes on a pipe with no one to read it.
SIGALRM	14	exit	Alarm clock.
SIGTERM	15	exit	Software termination signal.

NAME

adduser, **addgroup** – add a user or group to the system

SYNOPSIS

adduser [options] [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID] [--firstuid ID] [--lastuid ID] [--ingroup GROUP | --gid ID] [--disabled-password] [--disabled-login] [--gecos GECOS] [--add_extra_groups] [--encrypt-home] user

adduser --system [options] [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID] [--group | --ingroup GROUP | --gid ID] [--disabled-password] [--disabled-login] [--gecos GECOS] user

addgroup [options] [--gid ID] group

addgroup --system [options] [--gid ID] group

adduser [options] user group

COMMON OPTIONS

[--quiet] [--debug] [--force-badname] [--help|-h] [--version] [--conf FILE]

DESCRIPTION

adduser and **addgroup** add users and groups to the system according to command line options and configuration information in */etc/adduser.conf*. They are friendlier front ends to the low level tools like **useradd**, **groupadd** and **usermod** programs, by default choosing Debian policy conformant UID and GID values, creating a home directory with skeletal configuration, running a custom script, and other features. **adduser** and **addgroup** can be run in one of five modes:

Add a normal user

If called with one non-option argument and without the **--system** or **--group** options, **adduser** will add a normal user.

adduser will choose the first available UID from the range specified for normal users in the configuration file. The UID can be overridden with the **--uid** option.

The range specified in the configuration file may be overridden with the **--firstuid** and **--lastuid** options.

By default, each user in Debian GNU/Linux is given a corresponding group with the same name. User-groups allow group writable directories to be easily maintained by placing the appropriate users in the new group, setting the set-group-ID bit in the directory, and ensuring that all users use a umask of 002. If this option is turned off by setting **USERGROUPS** to *no*, all users' GIDs are set to **USERS_GID**. Users' primary groups can also be overridden from the command line with the **--gid** or **--ingroup** options to set the group by id or name, respectively. Also, users can be added to one or more groups defined in adduser.conf either by setting **ADD_EXTRA_GROUPS** to 1 in adduser.conf, or by passing **--add_extra_groups** on the commandline.

adduser will create a home directory subject to **DHOME**, **GROUPTHOMES**, and **LETTERHOMES**. The home directory can be overridden from the command line with the **--home** option, and the shell with the **--shell** option. The home directory's set-group-ID bit is set if **USERGROUPS** is *yes* so that any files created in the user's home directory will have the correct group.

adduser will copy files from **SKEL** into the home directory and prompt for finger (gecos) information and a password. The gecos may also be set with the **--gecos** option. With the **--disabled-login** option, the account will be created but will be disabled until a password is set. The **--disabled-password** option will not set a password, but login is still possible (for example with SSH RSA keys). To set up an encrypted home directory for the new user, add the **--encrypt-home** option. For more information, refer to the **-b** option of **cryptfs-setup-private(1)**.

If the file **/usr/local/sbin/adduser.local** exists, it will be executed after the user account has been set up in order to do any local setup. The arguments passed to **adduser.local** are:

username uid gid home-directory

The environment variable VERBOSE is set according to the following rule:

0 if **--quiet** is specified

1 if neither

--quiet nor **--debug** is specified

2 if **--debug** is specified

(The same applies to the variable DEBUG, but DEBUG is deprecated and will be removed in a later version of **adduser**.)

Add a system user

If called with one non-option argument and the **--system** option, **adduser** will add a system user. If a user with the same name already exists in the system uid range (or, if the uid is specified, if a user with that uid already exists), **adduser** will exit with a warning. This warning can be suppressed by adding **--quiet**.

adduser will choose the first available UID from the range specified for system users in the configuration file (FIRST_SYSTEM_UID and LAST_SYSTEM_UID). If you want to have a specific UID, you can specify it using the **--uid** option.

By default, system users are placed in the **nogroup** group. To place the new system user in an already existing group, use the **--gid** or **--ingroup** options. To place the new system user in a new group with the same ID, use the **--group** option.

A home directory is created by the same rules as for normal users. The new system user will have the shell */usr/sbin/nologin* (unless overridden with the **--shell** option), and have logins disabled. Skeletal configuration files are not copied.

Add a user group

If **adduser** is called with the **--group** option and without the **--system** option, or **addgroup** is called respectively, a user group will be added.

A GID will be chosen from the range specified for system GIDS in the configuration file (FIRST_GID, LAST_GID). To override that mechanism you can give the GID using the **--gid** option.

The group is created with no users.

Add a system group

If **addgroup** is called with the **--system** option, a system group will be added.

A GID will be chosen from the range specified for system GIDS in the configuration file (FIRST_SYSTEM_GID, LAST_SYSTEM_GID). To override that mechanism you can give the GID using the **--gid** option.

The group is created with no users.

Add an existing user to an existing group

If called with two non-option arguments, **adduser** will add an existing user to an existing group.

OPTIONS

--conf FILE

Use FILE instead of */etc/adduser.conf*.

--disabled-login

Do not run passwd to set the password. The user won't be able to use her account until the password is set.

--disabled-password

Like --disabled-login, but logins are still possible (for example using SSH RSA keys) but not using password authentication.

--force-badname

By default, user and group names are checked against the configurable regular expression **NAME_REGEX** (or **NAME_REGEX_SYSTEM** if --system is specified) specified in the configuration file. This option forces **adduser** and **addgroup** to apply only a weak check for validity of the name. **NAME_REGEX** is described in **adduser.conf(5)**.

--gecos GECOS

Set the gecos field for the new entry generated. **adduser** will not ask for finger information if this option is given.

--gid ID

When creating a group, this option forces the new groupid to be the given number. When creating a user, this option will put the user in that group.

--group

When combined with --system, a group with the same name and ID as the system user is created. If not combined with --system, a group with the given name is created. This is the default action if the program is invoked as **addgroup**.

--help Display brief instructions.**--home DIR**

Use DIR as the user's home directory, rather than the default specified by the configuration file. If the directory does not exist, it is created and skeleton files are copied.

--shell SHELL

Use SHELL as the user's login shell, rather than the default specified by the configuration file.

--ingroup GROUP

Add the new user to GROUP instead of a usergroup or the default group defined by **USERS_GID** in the configuration file. This affects the users primary group. To add additional groups, see the **add_extra_groups** option.

--no-create-home

Do not create the home directory, even if it doesn't exist.

--quiet

Suppress informational messages, only show warnings and errors.

--debug

Be verbose, most useful if you want to nail down a problem with adduser.

--system

Create a system user or group.

--uid ID

Force the new userid to be the given number. **adduser** will fail if the userid is already taken.

--firstuid ID

Override the first uid in the range that the uid is chosen from (overrides **FIRST_UID** specified in the configuration file).

--lastuid ID

Override the last uid in the range that the uid is chosen from (**LAST_UID**)

--add_extra_groups

Add new user to extra groups defined in the configuration file.

--version

Display version and copyright information.

EXIT VALUES

- 0** The user exists as specified. This can have 2 causes: The user was created by adduser or the user was already present on the system before adduser was invoked. If adduser was returning 0 , invoking adduser a second time with the same parameters as before also returns 0.
- 1** Creating the user or group failed because it was already present with other UID/GID than specified. The username or groupname was rejected because of a mismatch with the configured regular expressions, see adduser.conf(5). Adduser has been aborted by a signal.
Or for many other yet undocumented reasons which are printed to console then. You may then consider to remove **--quiet** to make adduser more verbose.

FILES

- /etc/adduser.conf
Default configuration file for adduser and addgroup
- /usr/local/sbin/adduser.local
Optional custom add-ons.

SEE ALSO

adduser.conf(5), deluser(8), groupadd(8), useradd(8), usermod(8), Debian Policy 9.2.2.

COPYRIGHT

Copyright (C) 1997, 1998, 1999 Guy Maor. Modifications by Roland Bauerschmidt and Marc Haber. Additional patches by Joerg Hoh and Stephen Gran.
Copyright (C) 1995 Ted Hajek, with a great deal borrowed from the original Debian **adduser**
Copyright (C) 1994 Ian Murdock. **adduser** is free software; see the GNU General Public Licence version 2 or later for copying conditions. There is *no* warranty.

NAME

adduser, **addgroup** – add a user or group to the system

SYNOPSIS

adduser [options] [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID] [--firstuid ID] [--lastuid ID] [--ingroup GROUP | --gid ID] [--disabled-password] [--disabled-login] [--gecos GECOS] [--add_extra_groups] [--encrypt-home] user

adduser --system [options] [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID] [--group | --ingroup GROUP | --gid ID] [--disabled-password] [--disabled-login] [--gecos GECOS] user

addgroup [options] [--gid ID] group

addgroup --system [options] [--gid ID] group

adduser [options] user group

COMMON OPTIONS

[--quiet] [--debug] [--force-badname] [--help|-h] [--version] [--conf FILE]

DESCRIPTION

adduser and **addgroup** add users and groups to the system according to command line options and configuration information in */etc/adduser.conf*. They are friendlier front ends to the low level tools like **useradd**, **groupadd** and **usermod** programs, by default choosing Debian policy conformant UID and GID values, creating a home directory with skeletal configuration, running a custom script, and other features. **adduser** and **addgroup** can be run in one of five modes:

Add a normal user

If called with one non-option argument and without the **--system** or **--group** options, **adduser** will add a normal user.

adduser will choose the first available UID from the range specified for normal users in the configuration file. The UID can be overridden with the **--uid** option.

The range specified in the configuration file may be overridden with the **--firstuid** and **--lastuid** options.

By default, each user in Debian GNU/Linux is given a corresponding group with the same name. User-groups allow group writable directories to be easily maintained by placing the appropriate users in the new group, setting the set-group-ID bit in the directory, and ensuring that all users use a umask of 002. If this option is turned off by setting **USERGROUPS** to *no*, all users' GIDs are set to **USERS_GID**. Users' primary groups can also be overridden from the command line with the **--gid** or **--ingroup** options to set the group by id or name, respectively. Also, users can be added to one or more groups defined in adduser.conf either by setting **ADD_EXTRA_GROUPS** to 1 in adduser.conf, or by passing **--add_extra_groups** on the commandline.

adduser will create a home directory subject to **DHOME**, **GROUPTHOMES**, and **LETTERHOMES**. The home directory can be overridden from the command line with the **--home** option, and the shell with the **--shell** option. The home directory's set-group-ID bit is set if **USERGROUPS** is *yes* so that any files created in the user's home directory will have the correct group.

adduser will copy files from **SKEL** into the home directory and prompt for finger (gecos) information and a password. The gecos may also be set with the **--gecos** option. With the **--disabled-login** option, the account will be created but will be disabled until a password is set. The **--disabled-password** option will not set a password, but login is still possible (for example with SSH RSA keys). To set up an encrypted home directory for the new user, add the **--encrypt-home** option. For more information, refer to the **-b** option of **cryptfs-setup-private(1)**.

If the file **/usr/local/sbin/adduser.local** exists, it will be executed after the user account has been set up in order to do any local setup. The arguments passed to **adduser.local** are:

username uid gid home-directory

The environment variable VERBOSE is set according to the following rule:

- 0 if **--quiet** is specified
- 1 if neither
 --quiet nor **--debug** is specified
- 2 if **--debug** is specified

(The same applies to the variable DEBUG, but DEBUG is deprecated and will be removed in a later version of **adduser**.)

Add a system user

If called with one non-option argument and the **--system** option, **adduser** will add a system user. If a user with the same name already exists in the system uid range (or, if the uid is specified, if a user with that uid already exists), **adduser** will exit with a warning. This warning can be suppressed by adding **--quiet**.

adduser will choose the first available UID from the range specified for system users in the configuration file (**FIRST_SYSTEM_UID** and **LAST_SYSTEM_UID**). If you want to have a specific UID, you can specify it using the **--uid** option.

By default, system users are placed in the **nogroup** group. To place the new system user in an already existing group, use the **--gid** or **--ingroup** options. To place the new system user in a new group with the same ID, use the **--group** option.

A home directory is created by the same rules as for normal users. The new system user will have the shell */usr/sbin/nologin* (unless overridden with the **--shell** option), and have logins disabled. Skeletal configuration files are not copied.

Add a user group

If **adduser** is called with the **--group** option and without the **--system** option, or **addgroup** is called respectively, a user group will be added.

A GID will be chosen from the range specified for system GIDS in the configuration file (**FIRST_GID**, **LAST_GID**). To override that mechanism you can give the GID using the **--gid** option.

The group is created with no users.

Add a system group

If **addgroup** is called with the **--system** option, a system group will be added.

A GID will be chosen from the range specified for system GIDS in the configuration file (**FIRST_SYSTEM_GID**, **LAST_SYSTEM_GID**). To override that mechanism you can give the GID using the **--gid** option.

The group is created with no users.

Add an existing user to an existing group

If called with two non-option arguments, **adduser** will add an existing user to an existing group.

OPTIONS

--conf FILE

Use FILE instead of */etc/adduser.conf*.

--disabled-login

Do not run *passwd* to set the password. The user won't be able to use her account until the password is set.

--disabled-password

Like --disabled-login, but logins are still possible (for example using SSH RSA keys) but not using password authentication.

--force-badname

By default, user and group names are checked against the configurable regular expression **NAME_REGEX** (or **NAME_REGEX_SYSTEM** if --system is specified) specified in the configuration file. This option forces **adduser** and **addgroup** to apply only a weak check for validity of the name. **NAME_REGEX** is described in **adduser.conf(5)**.

--gecos GECOS

Set the gecos field for the new entry generated. **adduser** will not ask for finger information if this option is given.

--gid ID

When creating a group, this option forces the new groupid to be the given number. When creating a user, this option will put the user in that group.

--group

When combined with --system, a group with the same name and ID as the system user is created. If not combined with --system, a group with the given name is created. This is the default action if the program is invoked as **addgroup**.

--help Display brief instructions.**--home DIR**

Use DIR as the user's home directory, rather than the default specified by the configuration file. If the directory does not exist, it is created and skeleton files are copied.

--shell SHELL

Use SHELL as the user's login shell, rather than the default specified by the configuration file.

--ingroup GROUP

Add the new user to GROUP instead of a usergroup or the default group defined by **USERS_GID** in the configuration file. This affects the users primary group. To add additional groups, see the **add_extra_groups** option.

--no-create-home

Do not create the home directory, even if it doesn't exist.

--quiet

Suppress informational messages, only show warnings and errors.

--debug

Be verbose, most useful if you want to nail down a problem with adduser.

--system

Create a system user or group.

--uid ID

Force the new userid to be the given number. **adduser** will fail if the userid is already taken.

--firstuid ID

Override the first uid in the range that the uid is chosen from (overrides **FIRST_UID** specified in the configuration file).

--lastuid ID

Override the last uid in the range that the uid is chosen from (**LAST_UID**)

--add_extra_groups

Add new user to extra groups defined in the configuration file.

--version

Display version and copyright information.

EXIT VALUES

- 0** The user exists as specified. This can have 2 causes: The user was created by adduser or the user was already present on the system before adduser was invoked. If adduser was returning 0 , invoking adduser a second time with the same parameters as before also returns 0.
- 1** Creating the user or group failed because it was already present with other UID/GID than specified. The username or groupname was rejected because of a mismatch with the configured regular expressions, see adduser.conf(5). Adduser has been aborted by a signal.
Or for many other yet undocumented reasons which are printed to console then. You may then consider to remove **--quiet** to make adduser more verbose.

FILES

- /etc/adduser.conf
Default configuration file for adduser and addgroup
- /usr/local/sbin/adduser.local
Optional custom add-ons.

SEE ALSO

adduser.conf(5), deluser(8), groupadd(8), useradd(8), usermod(8), Debian Policy 9.2.2.

COPYRIGHT

Copyright (C) 1997, 1998, 1999 Guy Maor. Modifications by Roland Bauerschmidt and Marc Haber. Additional patches by Joerg Hoh and Stephen Gran.
Copyright (C) 1995 Ted Hajek, with a great deal borrowed from the original Debian **adduser**
Copyright (C) 1994 Ian Murdock. **adduser** is free software; see the GNU General Public Licence version 2 or later for copying conditions. There is *no* warranty.

NAME

/etc/adduser.conf – configuration file for **adduser(8)** and **addgroup(8)**.

DESCRIPTION

The file */etc/adduser.conf* contains defaults for the programs **adduser(8)** , **addgroup(8)** , **deluser(8)** and **delgroup(8)**. Each line holds a single value pair in the form *option = value*. Double or single quotes are allowed around the value, as is whitespace around the equals sign. Comment lines must have a hash sign (#) in the first column.

The valid configuration options are:

DHELL

The login shell to be used for all new users. Defaults to */bin/bash*.

DHOME

The directory in which new home directories should be created. Defaults to */home*.

GROUPHOMES

If this is set to *yes*, the home directories will be created as */home/[groupname]/user*. Defaults to *no*.

LETTERHOMES

If this is set to *yes*, then the home directories created will have an extra directory inserted which is the first letter of the loginname. For example: */home/u/user*. Defaults to *no*.

SKEL The directory from which skeletal user configuration files should be copied. Defaults to */etc/skel*.

FIRST_SYSTEM_UID and LAST_SYSTEM_UID

specify an inclusive range of UIDs from which system UIDs can be dynamically allocated. Default to *100 - 999*. Please note that system software, such as the users allocated by the base-passwd package, may assume that UIDs less than 100 are unallocated.

FIRST_UID and LAST_UID

specify an inclusive range of UIDs from which normal user's UIDs can be dynamically allocated. Default to *1000 - 59999*.

FIRST_SYSTEM_GID and LAST_SYSTEM_GID

specify an inclusive range of GIDs from which system GIDs can be dynamically allocated. Default to *100 - 999*.

FIRST_GID and LAST_GID

specify an inclusive range of GIDs from which normal group's GIDs can be dynamically allocated. Default to *1000 - 59999*.

USERGROUPS

If this is set to *yes*, then each created user will be given their own group to use. If this is *no*, then each created user will be placed in the group whose GID is **USERS_GID** (see below). The default is *yes*.

USERS_GID

If **USERGROUPS** is *no*, then **USERS_GID** is the GID given to all newly-created users. The default value is *100*.

DIR_MODE

If set to a valid value (e.g. *0755* or *755*), directories created will have the specified permissions as umask. Otherwise *0755* is used as default.

SETGID_HOME

If this is set to *yes*, then home directories for users with their own group (*USERGR_OUPS=yes*) will have the setgid bit set. This was the default setting for adduser versions << 3.13. Unfortunately it has some bad side effects, so we no longer do this per default. If you want it nevertheless you can still activate it here.

QUOTAUSER

If set to a nonempty value, new users will have quotas copied from that user. The default is empty.

NAME_REGEX

User and group names are checked against this regular expression. If the name doesn't match this regexp, user and group creation in adduser is refused unless --force-badname is set. With --force-badname set, only weak checks are performed. The default is the most conservative ^[a-z][-a-z0-9]*\$. When --system is specified, NAME_REGEX_SYSTEM is used instead.

NAME_REGEX_SYSTEM

Names of system users are checked against this regular expression. If --system is supplied and the name doesn't match this regexp, user creation in adduser is refused unless --force-badname is set. With --force-badname set, only weak checks are performed. The default is as for the default NAME_REGEX but also allowing uppercase letters.

SKEL_IGNORE_REGEX

Files in /etc/skel/ are checked against this regex, and not copied to the newly created home directory if they match. This is by default set to the regular expression matching files left over from unmerged config files (dpkg-(old|new|dist)).

ADD_EXTRA_GROUPS

Setting this to something other than 0 (the default) will cause adduser to add newly created non-system users to the list of groups defined by EXTRA_GROUPS (below).

EXTRA_GROUPS

This is the list of groups that new non-system users will be added to. By default, this list is 'dialout cdrom floppy audio video plugdev users games'.

NOTES**VALID NAMES**

adduser and addgroup enforce conformity to IEEE Std 1003.1-2001, which allows only the following characters to appear in group and user names: letters, digits, underscores, periods, at signs (@) and dashes. The name may not start with a dash. The "\$" sign is allowed at the end of usernames (to conform to samba).

An additional check can be adjusted via the configuration parameter NAME_REGEX to enforce a local policy.

FILES

/etc/adduser.conf

SEE ALSO

addgroup(8), adduser(8), delgroup(8), deluser(8), deluser.conf(5)

NAME

apt-cache – query the APT cache

SYNOPSIS

```
apt-cache [-agipns] [-o=config_string] [-c=config_file] {gencaches | showpkg pkg... | showsrc pkg... |  
stats | dump | dumpavail | unmet | search regex... |  
show pkg [{=pkg_version_number} /target_release}...] |  
depends pkg [{=pkg_version_number} /target_release}...] |  
rdepends pkg [{=pkg_version_number} /target_release}...] | pkgnames [prefix] |  
dotty pkg [{=pkg_version_number} /target_release}...] |  
xvcg pkg [{=pkg_version_number} /target_release}...] | policy [pkg...] | madisonpkg... |  
{-v | --version} | {-h | --help}}
```

DESCRIPTION

apt-cache performs a variety of operations on APT's package cache. **apt-cache** does not manipulate the state of the system but does provide operations to search and generate interesting output from the package metadata. The metadata is acquired and updated via the 'update' command of e.g. **apt-get**, so that it can be outdated if the last update is too long ago, but in exchange **apt-cache** works independently of the availability of the configured sources (e.g. offline).

Unless the **-h**, or **--help** option is given, one of the commands below must be present.

gencaches

gencaches creates APT's package cache. This is done implicitly by all commands needing this cache if it is missing or outdated.

showpkg pkg...

showpkg displays information about the packages listed on the command line. Remaining arguments are package names. The available versions and reverse dependencies of each package listed are listed, as well as forward dependencies for each version. Forward (normal) dependencies are those packages upon which the package in question depends; reverse dependencies are those packages that depend upon the package in question. Thus, forward dependencies must be satisfied for a package, but reverse dependencies need not be. For instance, **apt-cache showpkg libreadline2** would produce output similar to the following:

```
Package: libreadline2
Versions: 2.1-12(/var/state/apt/lists/foo_Packages),
Reverse Depends:
    libreadline2,libreadline2
    libreadline2-altdev,libreadline2
Dependencies:
    2.1-12 - libc5 (2 5.4.0-0) ncurses3.0 (0 (null))
Provides:
    2.1-12 -
Reverse Provides:
Thus it may be seen that libreadline2, version 2.1-12, depends on libc5 and ncurses3.0 which must be installed for libreadline2 to work. In turn, libreadline2 and libreadline2-altdev depend on libreadline2. If libreadline2 is installed, libc5 and ncurses3.0 (and ldso) must also be installed; libreadline2 and libreadline2-altdev do not have to be installed. For the specific meaning of the remainder of the output it is best to consult the apt source code.
```

stats

stats displays some statistics about the cache. No further arguments are expected. Statistics reported are:

- Total package names is the number of package names found in the cache.
- Normal packages is the number of regular, ordinary package names; these are packages that bear a one-to-one correspondence between their names and the names used by other packages

for them in dependencies. The majority of packages fall into this category.

- Pure virtual packages is the number of packages that exist only as a virtual package name; that is, packages only "provide" the virtual package name, and no package actually uses the name. For instance, "mail-transport-agent" in the Debian system is a pure virtual package; several packages provide "mail-transport-agent", but there is no package named "mail-transport-agent".
- Single virtual packages is the number of packages with only one package providing a particular virtual package. For example, in the Debian system, "X11-text-viewer" is a virtual package, but only one package, xless, provides "X11-text-viewer".
- Mixed virtual packages is the number of packages that either provide a particular virtual package or have the virtual package name as the package name. For instance, in the Debian system, "debconf" is both an actual package, and provided by the debconf-tiny package.
- Missing is the number of package names that were referenced in a dependency but were not provided by any package. Missing packages may be an evidence if a full distribution is not accessed, or if a package (real or virtual) has been dropped from the distribution. Usually they are referenced from Conflicts or Breaks statements.
- Total distinct versions is the number of package versions found in the cache. If more than one distribution is being accessed (for instance, "stable" and "unstable"), this value can be considerably larger than the number of total package names.
- Total dependencies is the number of dependency relationships claimed by all of the packages in the cache.

showsdc *pkg...*

showsdc displays all the source package records that match the given package names. All versions are shown, as well as all records that declare the name to be a binary package. Use **--only-source** to display only source package names.

dump

dump shows a short listing of every package in the cache. It is primarily for debugging.

dumpavail

dumpavail prints out an available list to stdout. This is suitable for use with **dpkg(1)** and is used by the **dselect(1)** method.

unmet

unmet displays a summary of all unmet dependencies in the package cache.

show *pkg...*

show performs a function similar to **dpkg --print-avail**; it displays the package records for the named packages.

search *regex...*

search performs a full text search on all available package lists for the POSIX regex pattern given, see **regex(7)**. It searches the package names and the descriptions for an occurrence of the regular expression and prints out the package name and the short description, including virtual package names. If **--full** is given then output identical to show is produced for each matched package, and if **--names-only** is given then the long description is not searched, only the package name and provided packages are.

Separate arguments can be used to specify multiple search patterns that are and'ed together.

depends *pkg...*

depends shows a listing of each dependency a package has and all the possible other packages that can fulfill that dependency.

rdepends *pkg...*

`rdepends` shows a listing of each reverse dependency a package has.

pkgnames [prefix]

This command prints the name of each package APT knows. The optional argument is a prefix match to filter the name list. The output is suitable for use in a shell tab complete function and the output is generated extremely quickly. This command is best used with the `--generate` option.

Note that a package which APT knows of is not necessarily available to download, installable or installed, e.g. virtual packages are also listed in the generated list.

dotty pkg...

`dotty` takes a list of packages on the command line and generates output suitable for use by `dotty` from the [GraphViz](#)^[1] package. The result will be a set of nodes and edges representing the relationships between the packages. By default the given packages will trace out all dependent packages; this can produce a very large graph. To limit the output to only the packages listed on the command line, set the APT::Cache::GivenOnly option.

The resulting nodes will have several shapes; normal packages are boxes, pure virtual packages are triangles, mixed virtual packages are diamonds, missing packages are hexagons. Orange boxes mean recursion was stopped (leaf packages), blue lines are pre-depends, green lines are conflicts.

Caution, `dotty` cannot graph larger sets of packages.

xvcg pkg...

The same as `dotty`, only for `xvcg` from the [VCG tool](#)^[2].

policy [pkg...]

`policy` is meant to help debug issues relating to the preferences file. With no arguments it will print out the priorities of each source. Otherwise it prints out detailed information about the priority selection of the named package.

madison pkg...

apt-cache's `madison` command attempts to mimic the output format and a subset of the functionality of the Debian archive management tool, `madison`. It displays available versions of a package in a tabular format. Unlike the original `madison`, it can only display information for the architecture for which APT has retrieved package lists (APT::Architecture).

OPTIONS

All command line options may be set using the configuration file, the descriptions indicate the configuration option to set. For boolean options you can override the config file by using something like `-f`, `--no-f`, `-f=no` or several other variations.

-p, --pkg-cache

Select the file to store the package cache. The package cache is the primary cache used by all operations. Configuration Item: Dir::Cache::pkgcache.

-s, --src-cache

Select the file to store the source cache. The source is used only by gencaches and it stores a parsed version of the package information from remote sources. When building the package cache the source cache is used to avoid reparsing all of the package files. Configuration Item: Dir::Cache::srccache.

-q, --quiet

Quiet; produces output suitable for logging, omitting progress indicators. More q's will produce more quietness up to a maximum of 2. You can also use `-q=#` to set the quietness level, overriding the configuration file. Configuration Item: quiet.

-i, --important

Print only important dependencies; for use with `unmet` and `depends`. Causes only `Depends` and `Pre-Depends` relations to be printed. Configuration Item: APT::Cache::Important.

--no-pre-depends, --no-depends, --no-recommends, --no-suggests, --no-conflicts,

--no-breaks, --no-replaces, --no-enhances

Per default the **depends** and **rdepends** print all dependencies. This can be tweaked with these flags which will omit the specified dependency type. Configuration Item:
APT::Cache::ShowDependencyType e.g. APT::Cache::ShowRecommends.

--implicit

Per default **depends** and **rdepends** print only dependencies explicitly expressed in the metadata. With this flag it will also show dependencies implicitly added based on the encountered data. A Conflicts: foo e.g. expresses implicitly that this package also conflicts with the package foo from any other architecture. Configuration Item: APT::Cache::ShowImplicit.

-f, --full

Print full package records when searching. Configuration Item: APT::Cache::ShowFull.

-a, --all-versions

Print full records for all available versions. This is the default; to turn it off, use **--no-all-versions**. If **--no-all-versions** is specified, only the candidate version will be displayed (the one which would be selected for installation). This option is only applicable to the show command. Configuration Item: APT::Cache::AllVersions.

-g, --generate

Perform automatic package cache regeneration, rather than use the cache as it is. This is the default; to turn it off, use **--no-generate**. Configuration Item: APT::Cache::Generate.

--names-only, -n

Only search on the package and provided package names, not the long descriptions. Configuration Item: APT::Cache::NamesOnly.

--all-names

Make pkgnames print all names, including virtual packages and missing dependencies. Configuration Item: APT::Cache::AllNames.

--recurse

Make depends and rdepends recursive so that all packages mentioned are printed once. Configuration Item: APT::Cache::RecurseDepends.

--installed

Limit the output of depends and rdepends to packages which are currently installed. Configuration Item: APT::Cache::Installed.

--with-source *filename*

Adds the given file as a source for metadata. Can be repeated to add multiple files. Supported are currently *.deb, *.dsc, *.changes, Sources and Packages files as well as source package directories. Files are matched based on their name only, not their content!

Sources and Packages can be compressed in any format apt supports as long as they have the correct extension. If you need to store multiple of these files in one directory you can prefix a name of your choice with the last character being an underscore ("_"). Example: my.example_Packages.xz

Note that these sources are treated as trusted (see **apt-secure(8)**). Configuration Item:
APT::Sources::With.

-h, --help

Show a short usage summary.

-v, --version

Show the program version.

-c, --config-file

Configuration File; Specify a configuration file to use. The program will read the default configuration file and then this configuration file. If configuration settings need to be set before the default configuration files are parsed specify a file with the **APT_CONFIG** environment variable. See

apt.conf(5) for syntax information.

-o, --option

Set a Configuration Option; This will set an arbitrary configuration option. The syntax is **-o Foo::Bar=bar**. **-o** and **--option** can be used multiple times to set different options.

FILES

/etc/apt/sources.list

Locations to fetch packages from. Configuration Item: Dir::Etc::SourceList.

/etc/apt/sources.list.d/

File fragments for locations to fetch packages from. Configuration Item: Dir::Etc::SourceParts.

/var/lib/apt/lists/

Storage area for state information for each package resource specified in **sources.list(5)** Configuration Item: Dir::State::Lists.

/var/lib/apt/lists/partial/

Storage area for state information in transit. Configuration Item: Dir::State::Lists (partial will be implicitly appended)

SEE ALSO

apt.conf(5), sources.list(5), apt-get(8)

DIAGNOSTICS

apt-cache returns zero on normal operation, decimal 100 on error.

BUGS

[APT bug page](#)^[3]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the **reportbug(1)** command.

AUTHORS

Jason Gunthorpe

APT team

NOTES

1. GraphViz
<http://www.research.att.com/sw/tools/graphviz/>
2. VCG tool
<http://rw4.cs.uni-sb.de/users/sander/html/gsvcg1.html>
3. APT bug page
<http://bugs.debian.org/src:apt>

NAME

`apt-config` – APT Configuration Query program

SYNOPSIS

```
apt-config [--empty] [--format '%f "%v";%n'] [-o=config_string] [-c=config_file] { shell | dump |
    {-v | --version} | {-h | --help} }
```

DESCRIPTION

apt-config is an internal program used by various portions of the APT suite to provide consistent configurability. It accesses the main configuration file /etc/apt/apt.conf in a manner that is easy to use for scripted applications.

Unless the **-h**, or **--help** option is given, one of the commands below must be present.

shell

shell is used to access the configuration information from a shell script. It is given pairs of arguments, the first being a shell variable and the second the configuration value to query. As output it lists shell assignment commands for each value present. In a shell script it should be used as follows:

```
OPTS="-f"
RES='apt-config shell OPTS MyApp::options'
eval $RES
This will set the shell environment variable $OPTS to the value of MyApp::options with a default of -f.
```

The configuration item may be postfix with a /[fdbi]. f returns file names, d returns directories, b returns true or false and i returns an integer. Each of the returns is normalized and verified internally.

dump

Just show the contents of the configuration space.

OPTIONS

All command line options may be set using the configuration file, the descriptions indicate the configuration option to set. For boolean options you can override the config file by using something like **-f-**, **--no-f**, **-f=no** or several other variations.

--empty

Include options which have an empty value. This is the default, so use **--no-empty** to remove them from the output.

--format '%f "%v";%n'

Defines the output of each config option. %t will be replaced with its individual name, %f with its full hierarchical name and %v with its value. Use uppercase letters and special characters in the value will be encoded to ensure that it can e.g. be safely used in a quoted-string as defined by RFC822.

Additionally %n will be replaced by a newline, and %N by a tab. A % can be printed by using %%.

-h, --help

Show a short usage summary.

-v, --version

Show the program version.

-c, --config-file

Configuration File; Specify a configuration file to use. The program will read the default configuration file and then this configuration file. If configuration settings need to be set before the default configuration files are parsed specify a file with the **APT_CONFIG** environment variable. See **apt.conf(5)** for syntax information.

-o, --option

Set a Configuration Option; This will set an arbitrary configuration option. The syntax is **-o Foo::Bar=bar**. **-o** and **--option** can be used multiple times to set different options.

SEE ALSO

[apt.conf\(5\)](#)

DIAGNOSTICS

apt-config returns zero on normal operation, decimal 100 on error.

BUGS

[APT bug page](#)^[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the `reportbug(1)` command.

AUTHORS

Jason Gunthorpe

APT team

NOTES

1. APT bug page
<http://bugs.debian.org/src:apt>

NAME

apt-get – APT package handling utility — command-line interface

SYNOPSIS

```
apt-get [-asqdyfmubV] [-o=config_string] [-c=config_file] [-t=target_release] [-a=architecture]
  {update | upgrade | dselect-upgrade | dist-upgrade |
  install pkg [{=pkg_version_number | /target_release}]]... | remove pkg... | purge pkg... |
  source pkg [{=pkg_version_number | /target_release}]]... |
  build-dep pkg [{=pkg_version_number | /target_release}]]... |
  download pkg [{=pkg_version_number | /target_release}]]... | check | clean | autoclean |
  autoremove | {-v | --version} | {-h | --help}
```

DESCRIPTION

apt-get is the command-line tool for handling packages, and may be considered the user's "back-end" to other tools using the APT library. Several "front-end" interfaces exist, such as **aptitude**(8), **synaptic**(8) and **wajig**(1).

Unless the **-h**, or **--help** option is given, one of the commands below must be present.

update

update is used to resynchronize the package index files from their sources. The indexes of available packages are fetched from the location(s) specified in `/etc/apt/sources.list`. For example, when using a Debian archive, this command retrieves and scans the `Packages.gz` files, so that information about new and updated packages is available. An update should always be performed before an upgrade or `dist-upgrade`. Please be aware that the overall progress meter will be incorrect as the size of the package files cannot be known in advance.

upgrade

upgrade is used to install the newest versions of all packages currently installed on the system from the sources enumerated in `/etc/apt/sources.list`. Packages currently installed with new versions available are retrieved and upgraded; under no circumstances are currently installed packages removed, or packages not already installed retrieved and installed. New versions of currently installed packages that cannot be upgraded without changing the install status of another package will be left at their current version. An update must be performed first so that **apt-get** knows that new versions of packages are available.

dist-upgrade

dist-upgrade in addition to performing the function of `upgrade`, also intelligently handles changing dependencies with new versions of packages; **apt-get** has a "smart" conflict resolution system, and it will attempt to upgrade the most important packages at the expense of less important ones if necessary. The `dist-upgrade` command may therefore remove some packages. The `/etc/apt/sources.list` file contains a list of locations from which to retrieve desired package files. See also **apt_preferences**(5) for a mechanism for overriding the general settings for individual packages.

dselect-upgrade

dselect-upgrade is used in conjunction with the traditional Debian packaging front-end, **dselect**(1). **dselect-upgrade** follows the changes made by **dselect**(1) to the Status field of available packages, and performs the actions necessary to realize that state (for instance, the removal of old and the installation of new packages).

install

install is followed by one or more packages desired for installation or upgrading. Each package is a package name, not a fully qualified filename (for instance, in a Debian system, `apt-utils` would be the argument provided, not `apt-utils_2.4.8_amd64.deb`). All packages required by the package(s) specified for installation will also be retrieved and installed. The `/etc/apt/sources.list` file is used to locate the desired packages. If a hyphen is appended to the package name (with no intervening space), the identified package will be removed if it is installed. Similarly a plus sign can be used to designate a package to install. These latter features may be used to override decisions made by `apt-get`'s conflict resolution system.

A specific version of a package can be selected for installation by following the package name with an equals and the version of the package to select. This will cause that version to be located and selected for install. Alternatively a specific distribution can be selected by following the package name with a slash and the version of the distribution or the Archive name (stable, testing, unstable).

Both of the version selection mechanisms can downgrade packages and must be used with care.

This is also the target to use if you want to upgrade one or more already-installed packages without upgrading every package you have on your system. Unlike the "upgrade" target, which installs the newest version of all currently installed packages, "install" will install the newest version of only the package(s) specified. Simply provide the name of the package(s) you wish to upgrade, and if a newer version is available, it (and its dependencies, as described above) will be downloaded and installed.

Finally, the **apt_preferences**(5) mechanism allows you to create an alternative installation policy for individual packages.

If no package matches the given expression and the expression contains one of '.', '?' or '*' then it is assumed to be a POSIX regular expression, and it is applied to all package names in the database. Any matches are then installed (or removed). Note that matching is done by substring so 'lo.*' matches 'how-lo' and 'lowest'. If this is undesired, anchor the regular expression with a '^' or '\$' character, or create a more specific regular expression.

Fallback to regular expressions is deprecated in APT 2.0, has been removed in **apt**(8), except for anchored expressions, and will be removed from **apt-get**(8) in a future version. Use **apt-patterns**(5) instead.

reinstall

reinstall is an alias for **install --reinstall**.

remove

remove is identical to **install** except that packages are removed instead of installed. Note that removing a package leaves its configuration files on the system. If a plus sign is appended to the package name (with no intervening space), the identified package will be installed instead of removed.

purge

purge is identical to **remove** except that packages are removed and purged (any configuration files are deleted too).

source

source causes **apt-get** to fetch source packages. APT will examine the available packages to decide which source package to fetch. It will then find and download into the current directory the newest available version of that source package while respecting the default release, set with the option **APT::Default-Release**, the **-t** option or per package with the **pkg/release** syntax, if possible.

The arguments are interpreted as binary and source package names. See the **--only-source** option if you want to change that.

Source packages are tracked separately from binary packages via deb-src lines in the **sources.list**(5) file. This means that you will need to add such a line for each repository you want to get sources from; otherwise you will probably get either the wrong (too old/too new) source versions or none at all.

If the **--compile** option is specified then the package will be compiled to a binary .deb using **dpkg-buildpackage** for the architecture as defined by the **--host-architecture** option. If **--download-only** is specified then the source package will not be unpacked.

A specific source version can be retrieved by postfixing the source name with an equals and then the version to fetch, similar to the mechanism used for the package files. This enables exact matching of

the source package name and version, implicitly enabling the APT::Get::Only-Source option.

Note that source packages are not installed and tracked in the **dpkg** database like binary packages; they are simply downloaded to the current directory, like source tarballs.

build-dep

build-dep causes apt-get to install/remove packages in an attempt to satisfy the build dependencies for a source package. By default the dependencies are satisfied to build the package natively. If desired a host-architecture can be specified with the **--host-architecture** option instead.

The arguments are interpreted as binary or source package names. See the **--only-source** option if you want to change that.

satisfy

satisfy causes apt-get to satisfy the given dependency strings. The dependency strings may have build profiles and architecture restriction list as in build dependencies. They may optionally be prefixed with "Conflicts: " to unsatisfy the dependency string. Multiple strings of the same type can be specified.

Example: apt-get satisfy "foo" "Conflicts: bar" "baz (>> 1.0) | bar (= 2.0), moo"

The legacy operator '</>' is not supported, use '<=>' instead.

check

check is a diagnostic tool; it updates the package cache and checks for broken dependencies.

download

download will download the given binary package into the current directory.

clean

clean clears out the local repository of retrieved package files. It removes everything but the lock file from /var/cache/apt/archives/ and /var/cache/apt/archives/partial/.

autoclean (and the **auto-clean** alias since 1.1)

Like **clean**, **autoclean** clears out the local repository of retrieved package files. The difference is that it only removes package files that can no longer be downloaded, and are largely useless. This allows a cache to be maintained over a long period without it growing out of control. The configuration option APT::Clean-Installed will prevent installed packages from being erased if it is set to off.

autoremove (and the **auto-remove** alias since 1.1)

autoremove is used to remove packages that were automatically installed to satisfy dependencies for other packages and are now no longer needed.

changelog

changelog tries to download the changelog of a package and displays it through **sensible-pager**. By default it displays the changelog for the version that is installed. However, you can specify the same options as for the **install** command.

indextargets

Displays by default a deb822 formatted listing of information about all data files (aka index targets) **apt-get update** would download. Supports a **--format** option to modify the output format as well as accepts lines of the default output to filter the records by. The command is mainly used as an interface for external tools working with APT to get information as well as filenames for downloaded files so they can use them as well instead of downloading them again on their own. Detailed documentation is omitted here and can instead be found in the file /usr/share/doc/apt/acquire-additional-files.md.gz shipped by the apt-doc package.

OPTIONS

All command line options may be set using the configuration file, the descriptions indicate the configuration option to set. For boolean options you can override the config file by using something like **-f**, **--no-f**, **-f=no** or several other variations.

--no-install-recommends

Do not consider recommended packages as a dependency for installing. Configuration Item: APT::Install-Recommends.

--install-suggests

Consider suggested packages as a dependency for installing. Configuration Item: APT::Install-Suggests.

-d, --download-only

Download only; package files are only retrieved, not unpacked or installed. Configuration Item: APT::Get::Download-Only.

-f, --fix-broken

Fix; attempt to correct a system with broken dependencies in place. This option, when used with install/remove, can omit any packages to permit APT to deduce a likely solution. If packages are specified, these have to completely correct the problem. The option is sometimes necessary when running APT for the first time; APT itself does not allow broken package dependencies to exist on a system. It is possible that a system's dependency structure can be so corrupt as to require manual intervention (which usually means using **dpkg --remove** to eliminate some of the offending packages). Use of this option together with **-m** may produce an error in some situations.

Configuration Item: APT::Get::Fix-Broken.

-m, --ignore-missing, --fix-missing

Ignore missing packages; if packages cannot be retrieved or fail the integrity check after retrieval (corrupted package files), hold back those packages and handle the result. Use of this option together with **-f** may produce an error in some situations. If a package is selected for installation (particularly if it is mentioned on the command line) and it could not be downloaded then it will be silently held back. Configuration Item: APT::Get::Fix-Missing.

--no-download

Disables downloading of packages. This is best used with **--ignore-missing** to force APT to use only the .debs it has already downloaded. Configuration Item: APT::Get::Download.

-q, --quiet

Quiet; produces output suitable for logging, omitting progress indicators. More q's will produce more quiet up to a maximum of 2. You can also use **-q=#** to set the quiet level, overriding the configuration file. Note that quiet level 2 implies **-y**; you should never use **-qq** without a no-action modifier such as **-d, --print-uris** or **-s** as APT may decide to do something you did not expect. Configuration Item: quiet.

-s, --simulate, --just-print, --dry-run, --recon, --no-act

No action; perform a simulation of events that would occur based on the current system state but do not actually change the system. Locking will be disabled (**Debug::NoLocking**) so the system state could change while **apt-get** is running. Simulations can also be executed by non-root users which might not have read access to all apt configuration distorting the simulation. A notice expressing this warning is also shown by default for non-root users (**APT::Get::Show-User-Simulation-Note**). Configuration Item: APT::Get::Simulate.

Simulated runs print out a series of lines, each representing a **dpkg** operation: configure (Conf), remove (Remv) or unpack (Inst). Square brackets indicate broken packages, and empty square brackets indicate breaks that are of no consequence (rare).

-y, --yes, --assume-yes

Automatic yes to prompts; assume "yes" as answer to all prompts and run non-interactively. If an undesirable situation, such as changing a held package, trying to install an unauthenticated package or removing an essential package occurs then apt-get will abort. Configuration Item: APT::Get::Assume-Yes.

--assume-no

Automatic "no" to all prompts. Configuration Item: APT::Get::Assume-No.

--no-show-upgraded

Do not show a list of all packages that are to be upgraded. Configuration Item:
APT::Get::Show-Upgraded.

-V, --verbose-versions

Show full versions for upgraded and installed packages. Configuration Item:
APT::Get::Show-Versions.

-a, --host-architecture

This option controls the architecture packages are built for by **apt-get source --compile** and how cross-builddependencies are satisfied. By default is it not set which means that the host architecture is the same as the build architecture (which is defined by APT::Architecture). Configuration Item:
APT::Get::Host-Architecture.

-P, --build-profiles

This option controls the activated build profiles for which a source package is built by **apt-get source --compile** and how build dependencies are satisfied. By default no build profile is active. More than one build profile can be activated at a time by concatenating them with a comma. Configuration Item:
APT::Build-Profiles.

-b, --compile, --build

Compile source packages after downloading them. Configuration Item: APT::Get::Compile.

--ignore-hold

Ignore package holds; this causes **apt-get** to ignore a hold placed on a package. This may be useful in conjunction with dist-upgrade to override a large number of undesired holds. Configuration Item:
APT::Ignore-Hold.

--with-new-pkgs

Allow installing new packages when used in conjunction with upgrade. This is useful if the update of an installed package requires new dependencies to be installed. Instead of holding the package back upgrade will upgrade the package and install the new dependencies. Note that upgrade with this option will never remove packages, only allow adding new ones. Configuration Item:
APT::Get::Upgrade-Allow-New.

--no-upgrade

Do not upgrade packages; when used in conjunction with install, no-upgrade will prevent packages on the command line from being upgraded if they are already installed. Configuration Item:
APT::Get::Upgrade.

--only-upgrade

Do not install new packages; when used in conjunction with install, only-upgrade will install upgrades for already installed packages only and ignore requests to install new packages. Configuration Item:
APT::Get::Only-Upgrade.

--allow-downgrades

This is a dangerous option that will cause apt to continue without prompting if it is doing downgrades. It should not be used except in very special situations. Using it can potentially destroy your system! Configuration Item: APT::Get::allow-downgrades. Introduced in APT 1.1.

--allow-remove-essential

Force yes; this is a dangerous option that will cause apt to continue without prompting if it is removing essentials. It should not be used except in very special situations. Using it can potentially destroy your system! Configuration Item: APT::Get::allow-remove-essential. Introduced in APT 1.1.

--allow-change-held-packages

Force yes; this is a dangerous option that will cause apt to continue without prompting if it is changing held packages. It should not be used except in very special situations. Using it can potentially destroy your system! Configuration Item: APT::Get::allow-change-held-packages. Introduced in APT 1.1.

--force-yes

Force yes; this is a dangerous option that will cause apt to continue without prompting if it is doing

something potentially harmful. It should not be used except in very special situations. Using force-yes can potentially destroy your system! Configuration Item: APT::Get::force-yes. This is deprecated and replaced by **--allow-unauthenticated**, **--allow-downgrades**, **--allow-remove-essential**, **--allow-change-held-packages** in 1.1.

--print-uris

Instead of fetching the files to install their URIs are printed. Each URI will have the path, the destination file name, the size and the expected MD5 hash. Note that the file name to write to will not always match the file name on the remote site! This also works with the source and update commands. When used with the update command the MD5 and size are not included, and it is up to the user to decompress any compressed files. Configuration Item: APT::Get::Print-URIs.

--purge

Use purge instead of remove for anything that would be removed. An asterisk ("*") will be displayed next to packages which are scheduled to be purged. **remove --purge** is equivalent to the **purge** command. Configuration Item: APT::Get::Purge.

--reinstall

Re-install packages that are already installed and at the newest version. Configuration Item: APT::Get::ReInstall.

--list-cleanup

This option is on by default; use **--no-list-cleanup** to turn it off. When it is on, **apt-get** will automatically manage the contents of /var/lib/apt/lists to ensure that obsolete files are erased. The only reason to turn it off is if you frequently change your sources list. Configuration Item: APT::Get::List-Cleanup.

-t, --target-release, --default-release

This option controls the default input to the policy engine; it creates a default pin at priority 990 using the specified release string. This overrides the general settings in /etc/apt/preferences. Specifically pinned packages are not affected by the value of this option. In short, this option lets you have simple control over which distribution packages will be retrieved from. Some common examples might be **-t '2.1***, **-t unstable** or **-t sid**. Configuration Item: APT::Default-Release; see also the **apt_preferences(5)** manual page.

--trivial-only

Only perform operations that are 'trivial'. Logically this can be considered related to **--assume-yes**; where **--assume-yes** will answer yes to any prompt, **--trivial-only** will answer no. Configuration Item: APT::Get::Trivial-Only.

--mark-auto

After successful installation, mark all freshly installed packages as automatically installed, which will cause each of the packages to be removed when no more manually installed packages depend on this package. This is equally to running **apt-mark auto** for all installed packages. Configuration Item: APT::Get::Mark-Auto.

--no-remove

If any packages are to be removed apt-get immediately aborts without prompting. Configuration Item: APT::Get::Remove.

--auto-remove, --autoremove

If the command is either install or remove, then this option acts like running the autoremove command, removing unused dependency packages. Configuration Item: APT::Get::AutomaticRemove.

--only-source

Only has meaning for the source and build-dep commands. Indicates that the given source names are not to be mapped through the binary table. This means that if this option is specified, these commands will only accept source package names as arguments, rather than accepting binary package names and looking up the corresponding source package. Configuration Item: APT::Get::Only-Source.

--diff-only, --dsc-only, --tar-only

Download only the diff, dsc, or tar file of a source archive. Configuration Item: APT::Get::Diff-Only, APT::Get::Dsc-Only, and APT::Get::Tar-Only.

--arch-only

Only process architecture-dependent build-dependencies. Configuration Item: APT::Get::Arch-Only.

--indep-only

Only process architecture-independent build-dependencies. Configuration Item: APT::Get::Indep-Only.

--allow-unauthenticated

Ignore if packages can't be authenticated and don't prompt about it. This can be useful while working with local repositories, but is a huge security risk if data authenticity isn't ensured in another way by the user itself. The usage of the **Trusted** option for **sources.list(5)** entries should usually be preferred over this global override. Configuration Item: APT::Get::AllowUnauthenticated.

--no-allow-insecure-repositories

Forbid the update command to acquire unverifiable data from configured sources. APT will fail at the update command for repositories without valid cryptographically signatures. See also **apt-secure(8)** for details on the concept and the implications. Configuration Item:

Acquire::AllowInsecureRepositories.

--allow-releaseinfo-change

Allow the update command to continue downloading data from a repository which changed its information of the release contained in the repository indicating e.g a new major release. APT will fail at the update command for such repositories until the change is confirmed to ensure the user is prepared for the change. See also **apt-secure(8)** for details on the concept and configuration.

Specialist options (**--allow-releaseinfo-change-field**) exist to allow changes only for certain fields like origin, label, codename, suite, version and defaultpin. See also **apt_preferences(5)**. Configuration Item: Acquire::AllowReleaseInfoChange.

--show-progress

Show user friendly progress information in the terminal window when packages are installed, upgraded or removed. For a machine parseable version of this data see README.progress-reporting in the apt doc directory. Configuration Items: Dpkg::Progress and Dpkg::Progress-Fancy.

--with-source *filename*

Adds the given file as a source for metadata. Can be repeated to add multiple files. See **--with-source** description in **apt-cache(8)** for further details.

-eany, --error-on=any

Fail the update command if any error occurred, even a transient one.

-h, --help

Show a short usage summary.

-v, --version

Show the program version.

-c, --config-file

Configuration File; Specify a configuration file to use. The program will read the default configuration file and then this configuration file. If configuration settings need to be set before the default configuration files are parsed specify a file with the **APT_CONFIG** environment variable. See **apt.conf(5)** for syntax information.

-o, --option

Set a Configuration Option; This will set an arbitrary configuration option. The syntax is **-o Foo::Bar=bar**. **-o** and **--option** can be used multiple times to set different options.

FILES

/etc/apt/sources.list

Locations to fetch packages from. Configuration Item: Dir::Etc::SourceList.

/etc/apt/sources.list.d/

File fragments for locations to fetch packages from. Configuration Item: Dir::Etc::SourceParts.

/etc/apt/apt.conf

APT configuration file. Configuration Item: Dir::Etc::Main.

/etc/apt/apt.conf.d/

APT configuration file fragments. Configuration Item: Dir::Etc::Parts.

/etc/apt/preferences

Version preferences file. This is where you would specify "pinning", i.e. a preference to get certain packages from a separate source or from a different version of a distribution. Configuration Item: Dir::Etc::Preferences.

/etc/apt/preferences.d/

File fragments for the version preferences. Configuration Item: Dir::Etc::PreferencesParts.

/var/cache/apt/archives/

Storage area for retrieved package files. Configuration Item: Dir::Cache::Archives.

/var/cache/apt/archives/partial/

Storage area for package files in transit. Configuration Item: Dir::Cache::Archives (partial will be implicitly appended)

/var/lib/apt/lists/

Storage area for state information for each package resource specified in **sources.list(5)** Configuration Item: Dir::State::Lists.

/var/lib/apt/lists/partial/

Storage area for state information in transit. Configuration Item: Dir::State::Lists (partial will be implicitly appended)

SEE ALSO

apt-cache(8), **apt-cdrom(8)**, **dpkg(1)**, **sources.list(5)**, **apt.conf(5)**, **apt-config(8)**, **apt-secure(8)**, The APT User's guide in /usr/share/doc/apt-doc/, **apt_preferences(5)**, the APT Howto.

DIAGNOSTICS

apt-get returns zero on normal operation, decimal 100 on error.

BUGS

[APT bug page](#)^[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the **reportbug(1)** command.

AUTHORS

Jason Gunthorpe

APT team

NOTES

1. APT bug page
<http://bugs.debian.org/src:apt>

NAME

apt-mark – show, set and unset various settings for a package

SYNOPSIS

```
apt-mark {-f=filename | {auto | manual} pkg... | {showauto | showmanual} [pkg...] } | {-v | --version} |
{-h | --help}

apt-mark {hold | unhold | install | remove | purge} pkg... |
{showhold | showinstall | showremove | showpurge} [pkg...]
```

DESCRIPTION

apt-mark can be used as a unified front-end to set various settings for a package, such as marking a package as being automatically/manually installed or changing **dpkg** selections such as hold, install, deinstall and purge which are respected e.g. by **apt-get dselect-upgrade** or **aptitude**.

AUTOMATICALLY AND MANUALLY INSTALLED PACKAGES

When you request that a package is installed, and as a result other packages are installed to satisfy its dependencies, the dependencies are marked as being automatically installed, while the package you installed explicitly is marked as manually installed. Once an automatically installed package is no longer depended on by any manually installed package it is considered no longer needed and e.g. **apt-get** or **aptitude** will at least suggest removing them.

auto

auto is used to mark a package as being automatically installed, which will cause the package to be removed when no more manually installed packages depend on this package.

manual

manual is used to mark a package as being manually installed, which will prevent the package from being automatically removed if no other packages depend on it.

minimize-manual

minimize-manual is used to mark (transitive) dependencies of metapackages as automatically installed. This can be used after an installation for example, to minimize the number of manually installed packages; or continuously on systems managed by system configuration metapackages.

showauto

showauto is used to print a list of automatically installed packages with each package on a new line. All automatically installed packages will be listed if no package is given. If packages are given only those which are automatically installed will be shown.

showmanual

showmanual can be used in the same way as **showauto** except that it will print a list of manually installed packages instead.

Options**-f=*filename*, --file=*filename***

Read/Write package stats from the filename given with the parameter *filename* instead of from the default location, which is `extended_states` in the directory defined by the Configuration Item: `Dir::State`.

PREVENT CHANGES FOR A PACKAGE**hold**

hold is used to mark a package as held back, which will prevent the package from being automatically installed, upgraded or removed.

unhold

unhold is used to cancel a previously set hold on a package to allow all actions again.

showhold

showhold is used to print a list of packages on hold in the same way as for the other show commands.

SCHEDULE PACKAGES FOR INSTALL, REMOVE AND PURGE

Some front-ends like **apt-get dselect-upgrade** can be used to apply previously scheduled changes to the install state of packages. Such changes can be scheduled with the **install**, **remove** (also known as **deinstall**) and **purge** commands. Packages with a specific selection can be displayed with **showinstall**, **showremove** and **showpurge** respectively. More information about these so called dpkg selections can be found in **dpkg(1)**.

OPTIONS

-h, --help

Show a short usage summary.

-v, --version

Show the program version.

-c, --config-file

Configuration File; Specify a configuration file to use. The program will read the default configuration file and then this configuration file. If configuration settings need to be set before the default configuration files are parsed specify a file with the **APT_CONFIG** environment variable. See **apt.conf(5)** for syntax information.

-o, --option

Set a Configuration Option; This will set an arbitrary configuration option. The syntax is **-o Foo::Bar=bar**. **-o** and **--option** can be used multiple times to set different options.

FILES

/var/lib/apt/extended_states

Status list of auto-installed packages. Configuration Item: Dir::State::extended_states.

SEE ALSO

apt-get(8), aptitude(8), apt.conf(5)

DIAGNOSTICS

apt-mark returns zero on normal operation, non-zero on error.

BUGS

[APT bug page](#)^[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the **reportbug(1)** command.

AUTHORS

Mike O'Connor

APT team

NOTES

1. APT bug page
<http://bugs.debian.org/src:apt>

NAME

apt – command-line interface

SYNOPSIS

```
apt [-h] [-o=config_string] [-c=config_file] [-t=target_release] [-a=architecture] {list | search | show |
    update | install pkg [{=pkg_version_number | /target_release}]}... | remove pkg... | upgrade |
    full-upgrade | edit-sources | {-v | --version} | {-h | --help}}
```

DESCRIPTION

apt provides a high-level commandline interface for the package management system. It is intended as an end user interface and enables some options better suited for interactive usage by default compared to more specialized APT tools like **apt-get(8)** and **apt-cache(8)**.

Much like **apt** itself, its manpage is intended as an end user interface and as such only mentions the most used commands and options partly to not duplicate information in multiple places and partly to avoid overwhelming readers with a cornucopia of options and details.

update (apt-get(8))

update is used to download package information from all configured sources. Other commands operate on this data to e.g. perform package upgrades or search in and display details about all packages available for installation.

upgrade (apt-get(8))

upgrade is used to install available upgrades of all packages currently installed on the system from the sources configured via **sources.list(5)**. New packages will be installed if required to satisfy dependencies, but existing packages will never be removed. If an upgrade for a package requires the removal of an installed package the upgrade for this package isn't performed.

full-upgrade (apt-get(8))

full-upgrade performs the function of upgrade but will remove currently installed packages if this is needed to upgrade the system as a whole.

install, reinstall, remove, purge (apt-get(8))

Performs the requested action on one or more packages specified via **regex(7)**, **glob(7)** or exact match. The requested action can be overridden for specific packages by appending a plus (+) to the package name to install this package or a minus (-) to remove it.

A specific version of a package can be selected for installation by following the package name with an equals (=) and the version of the package to select. Alternatively the version from a specific release can be selected by following the package name with a forward slash (/) and codename (bullseye, bookworm, sid ...) or suite name (stable, testing, unstable). This will also select versions from this release for dependencies of this package if needed to satisfy the request.

Removing a package removes all packaged data, but leaves usually small (modified) user configuration files behind, in case the remove was an accident. Just issuing an installation request for the accidentally removed package will restore its function as before in that case. On the other hand you can get rid of these leftovers by calling **purge** even on already removed packages. Note that this does not affect any data or configuration stored in your home directory.

autoremove (apt-get(8))

autoremove is used to remove packages that were automatically installed to satisfy dependencies for other packages and are now no longer needed as dependencies changed or the package(s) needing them were removed in the meantime.

You should check that the list does not include applications you have grown to like even though they were once installed just as a dependency of another package. You can mark such a package as manually installed by using **apt-mark(8)**. Packages which you have installed explicitly via **install** are also never proposed for automatic removal.

satisfy (apt-get(8))

satisfy satisfies dependency strings, as used in Build-Depends. It also handles conflicts, by prefixing an argument with "Conflicts: ".

Example: apt satisfy "foo, bar (>= 1.0)" "Conflicts: baz, fuzz"

search (apt-cache(8))

search can be used to search for the given **regex(7)** term(s) in the list of available packages and display matches. This can e.g. be useful if you are looking for packages having a specific feature. If you are looking for a package including a specific file try **apt-file(1)**.

show (apt-cache(8))

Show information about the given package(s) including its dependencies, installation and download size, sources the package is available from, the description of the packages content and much more. It can e.g. be helpful to look at this information before allowing **apt(8)** to remove a package or while searching for new packages to install.

list

list is somewhat similar to **dkg-query --list** in that it can display a list of packages satisfying certain criteria. It supports **glob(7)** patterns for matching package names as well as options to list installed (**--installed**), upgradeable (**--upgradeable**) or all available (**--all-versions**) versions.

edit-sources (work-in-progress)

edit-sources lets you edit your **sources.list(5)** files in your preferred text editor while also providing basic sanity checks.

SCRIPT USAGE AND DIFFERENCES FROM OTHER APT TOOLS

The **apt(8)** commandline is designed as an end-user tool and it may change behavior between versions. While it tries not to break backward compatibility this is not guaranteed either if a change seems beneficial for interactive use.

All features of **apt(8)** are available in dedicated APT tools like **apt-get(8)** and **apt-cache(8)** as well. **apt(8)** just changes the default value of some options (see **apt.conf(5)** and specifically the Binary scope). So you should prefer using these commands (potentially with some additional options enabled) in your scripts as they keep backward compatibility as much as possible.

SEE ALSO

apt-get(8), **apt-cache(8)**, **sources.list(5)**, **apt.conf(5)**, **apt-config(8)**, The APT User's guide in /usr/share/doc/apt-doc/, **apt_preferences(5)**, the APT Howto.

DIAGNOSTICS

apt returns zero on normal operation, decimal 100 on error.

BUGS

[APT bug page](#)^[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the **reportbug(1)** command.

AUTHOR

APT team

NOTES

1. APT bug page
<http://bugs.debian.org/src:apt>

NAME

apt.conf – Configuration file for APT

DESCRIPTION

/etc/apt/apt.conf is the main configuration file shared by all the tools in the APT suite of tools, though it is by no means the only place options can be set. The suite also shares a common command line parser to provide a uniform environment.

When an APT tool starts up it will read the configuration files in the following order:

1. the file specified by the **APT_CONFIG** environment variable (if any)
2. all files in Dir::Etc::Parts in alphanumeric ascending order which have either no or "conf" as filename extension and which only contain alphanumeric, hyphen (-), underscore (_) and period (.) characters. Otherwise APT will print a notice that it has ignored a file, unless that file matches a pattern in the Dir::Ignore-Files-Silently configuration list – in which case it will be silently ignored.
3. the main configuration file specified by Dir::Etc::main
4. all options set in the binary specific configuration subtree are moved into the root of the tree.
5. the command line options are applied to override the configuration directives or to load even more configuration files.

SYNTAX

The configuration file is organized in a tree with options organized into functional groups. Option specification is given with a double colon notation; for instance APT::Get::Assume-Yes is an option within the APT tool group, for the Get tool. Options do not inherit from their parent groups.

Syntactically the configuration language is modeled after what the ISC tools such as bind and dhcp use. Lines starting with // are treated as comments (ignored), as well as all text between /* and */, just like C/C++ comments. Lines starting with # are also treated as comments. Each line is of the form APT::Get::Assume-Yes "true";. The quotation marks and trailing semicolon are required. The value must be on one line, and there is no kind of string concatenation. Values must not include backslashes or extra quotation marks. Option names are made up of alphanumeric characters and the characters "/-:_+". A new scope can be opened with curly braces, like this:

```
APT {
  Get {
    Assume-Yes "true";
    Fix-Broken "true";
  };
};
```

with newlines placed to make it more readable. Lists can be created by opening a scope and including a single string enclosed in quotes followed by a semicolon. Multiple entries can be included, separated by a semicolon.

```
DPkg::Pre-Install-Pkgs {"/usr/sbin/dpkg-preconfigure --apt";};
```

In general the sample configuration file /usr/share/doc/apt/examples/configure-index.gz is a good guide for how it should look.

Case is not significant in names of configuration items, so in the previous example you could use dpkg::pre-install-pkgs.

Names for the configuration items are optional if a list is defined as can be seen in the DPkg::Pre-Install-Pkgs example above. If you don't specify a name a new entry will simply add a new option to the list. If you specify a name you can override the option in the same way as any other option by

reassigning a new value to the option.

Two special commands are defined: #include (which is deprecated and not supported by alternative implementations) and #clear. #include will include the given file, unless the filename ends in a slash, in which case the whole directory is included. #clear is used to erase a part of the configuration tree. The specified element and all its descendants are erased. (Note that these lines also need to end with a semicolon.)

The #clear command is the only way to delete a list or a complete scope. Reopening a scope (or using the syntax described below with an appended ::) will *not* override previously written entries. Options can only be overridden by addressing a new value to them – lists and scopes can't be overridden, only cleared.

All of the APT tools take an –o option which allows an arbitrary configuration directive to be specified on the command line. The syntax is a full option name (APT::Get::Assume-Yes for instance) followed by an equals sign then the new value of the option. To append a new element to a list, add a trailing :: to the name of the list. (As you might suspect, the scope syntax can't be used on the command line.)

Note that appending items to a list using :: only works for one item per line, and that you should not use it in combination with the scope syntax (which adds :: implicitly). Using both syntaxes together will trigger a bug which some users unfortunately depend on: an option with the unusual name "::" which acts like every other option with a name. This introduces many problems; for one thing, users who write multiple lines in this *wrong* syntax in the hope of appending to a list will achieve the opposite, as only the last assignment for this option "::" will be used. Future versions of APT will raise errors and stop working if they encounter this misuse, so please correct such statements now while APT doesn't explicitly complain about them.

THE APT GROUP

This group of options controls general APT behavior as well as holding the options for all of the tools.

Architecture

System Architecture; sets the architecture to use when fetching files and parsing package lists. The internal default is the architecture apt was compiled for.

Architectures

All Architectures the system supports. For instance, CPUs implementing the amd64 (also called x86-64) instruction set are also able to execute binaries compiled for the i386 (x86) instruction set. This list is used when fetching files and parsing package lists. The initial default is always the system's native architecture (APT::Architecture), and foreign architectures are added to the default list when they are registered via **dpkg --add-architecture**.

Compressor

This scope defines which compression formats are supported, how compression and decompression can be performed if support for this format isn't built into apt directly and a cost-value indicating how costly it is to compress something in this format. As an example the following configuration stanza would allow apt to download and uncompress as well as create and store files with the low-cost .reversed file extension which it will pass to the command **rev** without additional commandline parameters for compression and uncompression:

```
APT::Compressor::rev {
    Name "rev";
    Extension ".reversed";
    Binary "rev";
    CompressArg {};
    UncompressArg {};
    Cost "10";
};
```

Build-Profiles

List of all build profiles enabled for build-dependency resolution, without the "profile." namespace prefix. By default this list is empty. The **DEB_BUILD_PROFILES** as used by **dpkg-buildpackage(1)** overrides the list notation.

Default–Release

Default release to install packages from if more than one version is available. Contains release name, codename or release version. Examples: 'stable', 'testing', 'unstable', 'bullseye', 'bookworm', '4.0', '5.0*'. See also `apt_preferences(5)`.

Ignore–Hold

Ignore held packages; this global option causes the problem resolver to ignore held packages in its decision making.

Clean–Installed

Defaults to on. When turned on the autoclean feature will remove any packages which can no longer be downloaded from the cache. If turned off then packages that are locally installed are also excluded from cleaning – but note that APT provides no direct means to reinstall them.

Immediate–Configure

Defaults to on, which will cause APT to install essential and important packages as soon as possible in an install/upgrade operation, in order to limit the effect of a failing `dpkg(1)` call. If this option is disabled, APT treats an important package in the same way as an extra package: between the unpacking of the package A and its configuration there can be many other unpack or configuration calls for other unrelated packages B, C etc. If these cause the `dpkg(1)` call to fail (e.g. because package B's maintainer scripts generate an error), this results in a system state in which package A is unpacked but unconfigured – so any package depending on A is now no longer guaranteed to work, as its dependency on A is no longer satisfied.

The immediate configuration marker is also applied in the potentially problematic case of circular dependencies, since a dependency with the immediate flag is equivalent to a Pre–Dependency. In theory this allows APT to recognise a situation in which it is unable to perform immediate configuration, abort, and suggest to the user that the option should be temporarily deactivated in order to allow the operation to proceed. Note the use of the word "theory" here; in the real world this problem has rarely been encountered, in non–stable distribution versions, and was caused by wrong dependencies of the package in question or by a system in an already broken state; so you should not blindly disable this option, as the scenario mentioned above is not the only problem it can help to prevent in the first place.

Before a big operation like `dist–upgrade` is run with this option disabled you should try to explicitly install the package APT is unable to configure immediately; but please make sure you also report your problem to your distribution and to the APT team with the bug link below, so they can work on improving or correcting the upgrade process.

Force–LoopBreak

Never enable this option unless you *really* know what you are doing. It permits APT to temporarily remove an essential package to break a Conflicts/Conflicts or Conflicts/Pre–Depends loop between two essential packages. *Such a loop should never exist and is a grave bug.* This option will work if the essential packages are not `tar`, `gzip`, `libc`, `dpkg`, `dash` or anything that those packages depend on.

Cache–Start, Cache–Grow, Cache–Limit

APT uses since version 0.7.26 a resizable memory mapped cache file to store the available information. Cache–Start acts as a hint of the size the cache will grow to, and is therefore the amount of memory APT will request at startup. The default value is 20971520 bytes (~20 MB). Note that this amount of space needs to be available for APT; otherwise it will likely fail ungracefully, so for memory restricted devices this value should be lowered while on systems with a lot of configured sources it should be increased. Cache–Grow defines in bytes with the default of 1048576 (~1 MB) how much the cache size will be increased in the event the space defined by Cache–Start is not enough. This value will be applied again and again until either the cache is big enough to store all information or the size of the cache reaches the Cache–Limit. The default of Cache–Limit is 0 which stands for no limit. If Cache–Grow is set to 0 the automatic growth of the cache is disabled.

Build–Essential

Defines which packages are considered essential build dependencies.

Get

The Get subsection controls the **apt-get(8)** tool; please see its documentation for more information about the options here.

Cache

The Cache subsection controls the **apt-cache(8)** tool; please see its documentation for more information about the options here.

CDROM

The CDROM subsection controls the **apt-cdrom(8)** tool; please see its documentation for more information about the options here.

THE ACQUIRE GROUP

The Acquire group of options controls the download of packages as well as the various "acquire methods" responsible for the download itself (see also **sources.list(5)**).

Check–Date

Security related option defaulting to true, enabling time-related checks. Disabling it means that the machine's time cannot be trusted, and APT will hence disable all time-related checks, such as **Check–Valid–Until** and verifying that the Date field of a release file is not in the future.

Max–FutureTime

Maximum time (in seconds) before its creation (as indicated by the Date header) that the Release file should be considered valid. The default value is 10. Archive specific settings can be made by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list(5)** entries by using the **Date–Max–Future** option there.

Check–Valid–Until

Security related option defaulting to true, as giving a Release file's validation an expiration date prevents replay attacks over a long timescale, and can also for example help users to identify mirrors that are no longer updated – but the feature depends on the correctness of the clock on the user system. Archive maintainers are encouraged to create Release files with the Valid–Until header, but if they don't or a stricter value is desired the Max–ValidTime option below can be used. The **Check–Valid–Until** option of **sources.list(5)** entries should be preferred to disable the check selectively instead of using this global override.

Max–ValidTime

Maximum time (in seconds) after its creation (as indicated by the Date header) that the Release file should be considered valid. If the Release file itself includes a Valid–Until header the earlier date of the two is used as the expiration date. The default value is 0 which stands for "valid forever". Archive specific settings can be made by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list(5)** entries by using the **Valid–Until–Max** option there.

Min–ValidTime

Minimum time (in seconds) after its creation (as indicated by the Date header) that the Release file should be considered valid. Use this if you need to use a seldom updated (local) mirror of a more frequently updated archive with a Valid–Until header instead of completely disabling the expiration date checking. Archive specific settings can and should be used by appending the label of the archive to the option name. Preferably, the same can be achieved for specific **sources.list(5)** entries by using the **Valid–Until–Min** option there.

AllowTLS

Allow use of the internal TLS support in the http method. If set to false, this completely disables support for TLS in apt's own methods (excluding the curl-based https method). No TLS-related functions will be called anymore.

PDiff

Try to download deltas called PDiff for indexes (like Packages files) instead of downloading whole ones. True by default. Preferably, this can be set for specific **sources.list(5)** entries or index files by

using the **PDiff**s option there.

Two sub-options to limit the use of PDiff are also available: FileLimit can be used to specify a maximum number of PDiff files should be downloaded to update a file. SizeLimit on the other hand is the maximum percentage of the size of all patches compared to the size of the targeted file. If one of these limits is exceeded the complete file is downloaded instead of the patches.

By-Hash

Try to download indexes via an URI constructed from a hashsum of the expected file rather than downloaded via a well-known stable filename. True by default, but automatically disabled if the source indicates no support for it. Usage can be forced with the special value "force". Preferably, this can be set for specific **sources.list**(5) entries or index files by using the **By-Hash** option there.

Queue-Mode

Queuing mode; Queue-Mode can be one of host or access which determines how APT parallelizes outgoing connections. host means that one connection per target host will be opened, access means that one connection per URI type will be opened.

Retries

Number of retries to perform. If this is non-zero APT will retry failed files the given number of times.

Source-Symlinks

Use symlinks for source archives. If set to true then source archives will be symlinked when possible instead of copying. True is the default.

http https

The options in these scopes configure APT's acquire transports for the protocols HTTP and HTTPS and are documented in the **apt-transport-http**(1) and **apt-transport-https**(1) manpages respectively.

ftp

ftp::Proxy sets the default proxy to use for FTP URIs. It is in the standard form of `ftp://[[:user][[:pass]@]host[:port]/]`. Per host proxies can also be specified by using the form `ftp::Proxy::<host>` with the special keyword DIRECT meaning to use no proxies. If no one of the above settings is specified, **ftp_proxy** environment variable will be used. To use an FTP proxy you will have to set the `ftp::ProxyLogin` script in the configuration file. This entry specifies the commands to send to tell the proxy server what to connect to. Please see `/usr/share/doc/apt/examples/configure-index.gz` for an example of how to do this. The substitution variables representing the corresponding URI component are `$(PROXY_USER)`, `$(PROXY_PASS)`, `$(SITE_USER)`, `$(SITE_PASS)`, `$(SITE)` and `$(SITE_PORT)`.

The option timeout sets the timeout timer used by the method; this value applies to the connection as well as the data timeout.

Several settings are provided to control passive mode. Generally it is safe to leave passive mode on; it works in nearly every environment. However, some situations require that passive mode be disabled and port mode FTP used instead. This can be done globally or for connections that go through a proxy or for a specific host (see the sample config file for examples).

It is possible to proxy FTP over HTTP by setting the **ftp_proxy** environment variable to an HTTP URL – see the discussion of the http method above for syntax. You cannot set this in the configuration file and it is not recommended to use FTP over HTTP due to its low efficiency.

The setting ForceExtended controls the use of RFC2428 EPSV and EPRT commands. The default is false, which means these commands are only used if the control connection is IPv6. Setting this to true forces their use even on IPv4 connections. Note that most FTP servers do not support RFC2428.

cdrom

For URIs using the cdrom method, the only configurable option is the mount point, `cdrom::Mount`, which must be the mount point for the CD-ROM (or DVD, or whatever) drive as specified in

/etc/fstab. It is possible to provide alternate mount and unmount commands if your mount point cannot be listed in the fstab. The syntax is to put

```
/cdrom/:Mount "foo";
```

within the cdrom block. It is important to have the trailing slash. Unmount commands can be specified using UMount.

gpgv

For GPGV URIs the only configurable option is gpgv::Options, which passes additional parameters to gpgv.

CompressionTypes

List of compression types which are understood by the acquire methods. Files like Packages can be available in various compression formats. By default the acquire methods can decompress and recompress many common formats like **xz** and **gzip**; with this scope the supported formats can be queried, modified as well as support for more formats added (see also **APT::Compressor**). The syntax for this is:

```
Acquire::CompressionTypes::FileExtension "Methodname";
```

Also, the Order subgroup can be used to define in which order the acquire system will try to download the compressed files. The acquire system will try the first and proceed with the next compression type in this list on error, so to prefer one over the other type simply add the preferred type first – types not already added will be implicitly appended to the end of the list, so e.g.

```
Acquire::CompressionTypes::Order:: "gz";
```

can be used to prefer **gzip** compressed files over all other compression formats. If **xz** should be preferred over **gzip** and **bzip2** the configure setting should look like this:

```
Acquire::CompressionTypes::Order { "xz"; "gz"; };
```

It is not needed to add bz2 to the list explicitly as it will be added automatically.

Note that the Dir::Bin::*Methodname* will be checked at run time. If this option has been set and support for this format isn't directly built into apt, the method will only be used if this file exists; e.g. for the bzip2 method (the inbuilt) setting is:

```
Dir::Bin::bzip2 "/bin/bzip2";
```

Note also that list entries specified on the command line will be added at the end of the list specified in the configuration files, but before the default entries. To prefer a type in this case over the ones specified in the configuration files you can set the option direct – not in list style. This will not override the defined list; it will only prefix the list with this type.

The special type uncompressed can be used to give uncompressed files a preference, but note that most archives don't provide uncompressed files so this is mostly only usable for local mirrors.

GzipIndexes

When downloading gzip compressed indexes (Packages, Sources, or Translations), keep them gzip compressed locally instead of unpacking them. This saves quite a lot of disk space at the expense of more CPU requirements when building the local package caches. False by default.

Languages

The Languages subsection controls which Translation files are downloaded and in which order APT tries to display the description-translations. APT will try to display the first available description in

the language which is listed first. Languages can be defined with their short or long language codes. Note that not all archives provide Translation files for every language – the long language codes are especially rare.

The default list includes "environment" and "en". "environment" has a special meaning here: it will be replaced at runtime with the language codes extracted from the LC_MESSAGES environment variable. It will also ensure that these codes are not included twice in the list. If LC_MESSAGES is set to "C" only the Translation-en file (if available) will be used. To force APT to use no Translation file use the setting Acquire::Languages=none. "none" is another special meaning code which will stop the search for a suitable Translation file. This tells APT to download these translations too, without actually using them unless the environment specifies the languages. So the following example configuration will result in the order "en, de" in an English locale or "de, en" in a German one. Note that "fr" is downloaded, but not used unless APT is used in a French locale (where the order would be "fr, de, en").

```
Acquire::Languages { "environment"; "de"; "en"; "none"; "fr"; };
```

Note: To prevent problems resulting from APT being executed in different environments (e.g. by different users or by other programs) all Translation files which are found in /var/lib/apt/lists/ will be added to the end of the list (after an implicit "none").

ForceIPv4

When downloading, force to use only the IPv4 protocol.

ForceIPv6

When downloading, force to use only the IPv6 protocol.

MaxReleaseFileSize

The maximum file size of Release/Release.gpg/InRelease files. The default is 10MB.

EnableSrvRecords

This option controls if apt will use the DNS SRV server record as specified in RFC 2782 to select an alternative server to connect to. The default is "true".

AllowInsecureRepositories

Allow update operations to load data files from repositories without sufficient security information. The default value is "false". Concept, implications as well as alternatives are detailed in [apt-secure\(8\)](#).

AllowWeakRepositories

Allow update operations to load data files from repositories which provide security information, but these are deemed no longer cryptographically strong enough. The default value is "false". Concept, implications as well as alternatives are detailed in [apt-secure\(8\)](#).

AllowDowngradeToInsecureRepositories

Allow that a repository that was previously gpg signed to become unsigned during an update operation. When there is no valid signature for a previously trusted repository apt will refuse the update. This option can be used to override this protection. You almost certainly never want to enable this. The default is false. Concept, implications as well as alternatives are detailed in [apt-secure\(8\)](#).

Changelogs::URI scope

Acquiring changelogs can only be done if an URI is known from where to get them. Preferable the Release file indicates this in a 'Changelogs' field. If this isn't available the Label/Origin field of the Release file is used to check if a Acquire::Changelogs::URI::Label::*LABEL* or Acquire::Changelogs::URI::Origin::*ORIGIN* option exists and if so this value is taken. The value in the Release file can be overridden with Acquire::Changelogs::URI::Override::Label::*LABEL* or Acquire::Changelogs::URI::Override::Origin::*ORIGIN*. The value should be a normal URI to a text file, except that package specific data is replaced with the placeholder @CHANGEPATH@. The value for it is: 1. if the package is from a component (e.g. main) this is the first part otherwise it is omitted, 2. the first letter of source package name, except if the source package name starts with 'lib' in which

case it will be the first four letters. 3. The complete source package name. 4. the complete name again and 5. the source version. The first (if present), second, third and fourth part are separated by a slash ('/') and between the fourth and fifth part is an underscore ('_'). The special value 'no' is available for this option indicating that this source can't be used to acquire changelog files from. Another source will be tried if available in this case.

BINARY SPECIFIC CONFIGURATION

Especially with the introduction of the **apt** binary it can be useful to set certain options only for a specific binary as even options which look like they would effect only a certain binary like

APT::Get::Show-Versions effect **apt-get** as well as **apt**.

Setting an option for a specific binary only can be achieved by setting the option inside the **Binary::specific-binary** scope. Setting the option **APT::Get::Show-Versions** for the **apt** only can e.g. by done by setting **Binary::apt::APT::Get::Show-Versions** instead.

Note that as seen in the DESCRIPTION section further above you can't set binary-specific options on the commandline itself nor in configuration files loaded via the commandline.

DIRECTORIES

The **Dir::State** section has directories that pertain to local state information. **lists** is the directory to place downloaded package lists in and **status** is the name of the **dpkg(1)** status file. **preferences** is the name of the APT preferences file. **Dir::State** contains the default directory to prefix on all sub-items if they do not start with / or ../../

Dir::Cache contains locations pertaining to local cache information, such as the two package caches **srcpkgcache** and **pkgcache** as well as the location to place downloaded archives, **Dir::Cache::archives**. Generation of caches can be turned off by setting **pkgcache** or **srcpkgcache** to "". This will slow down startup but save disk space. It is probably preferable to turn off the **pkgcache** rather than the **srcpkgcache**. Like **Dir::State** the default directory is contained in **Dir::Cache**

Dir::Etc contains the location of configuration files, **sourcelist** gives the location of the sourcelist and **main** is the default configuration file (setting has no effect, unless it is done from the config file specified by **APT_CONFIG**).

The **Dir::Parts** setting reads in all the config fragments in lexical order from the directory specified. After this is done then the main config file is loaded.

Binary programs are pointed to by **Dir::Bin**. **Dir::Bin::Methods** specifies the location of the method handlers and gzip, bzip2, lzma, dpkg, apt-get dpkg-source dpkg-buildpackage and apt-cache specify the location of the respective programs.

The configuration item **RootDir** has a special meaning. If set, all paths will be relative to **RootDir**, *even paths that are specified absolutely*. So, for instance, if **RootDir** is set to /tmp/staging and **Dir::State::status** is set to /var/lib/dpkg/status, then the status file will be looked up in /tmp/staging/var/lib/dpkg/status. If you want to prefix only relative paths, set **Dir** instead.

The **Ignore-Files-Silently** list can be used to specify which files APT should silently ignore while parsing the files in the fragment directories. Per default a file which ends with .disabled, ~, .bak or .dpkg-[a-z]+ is silently ignored. As seen in the last default value these patterns can use regular expression syntax.

APT IN DSELECT

When APT is used as a **dselect(1)** method several configuration directives control the default behavior. These are in the DSelect section.

Clean

Cache Clean mode; this value may be one of always, prompt, auto, pre-auto and never. always and prompt will remove all packages from the cache after upgrading, prompt (the default) does so conditionally. auto removes only those packages which are no longer downloadable (replaced with a new version for instance). pre-auto performs this action before downloading new packages.

options

The contents of this variable are passed to **apt-get(8)** as command line options when it is run for the

install phase.

Updateoptions

The contents of this variable are passed to **apt-get(8)** as command line options when it is run for the update phase.

PromptAfterUpdate

If true the [U]date operation in **dselect(1)** will always prompt to continue. The default is to prompt only on error.

HOW APT CALLS DPKG(1)

Several configuration directives control how APT invokes **dpkg(1)**. These are in the DPkg section.

options

This is a list of options to pass to **dpkg(1)**. The options must be specified using the list notation and each list item is passed as a single argument to **dpkg(1)**.

Path

This is a string that defines the **PATH** environment variable used when running dpkg. It may be set to any valid value of that environment variable; or the empty string, in which case the variable is not changed.

Pre-Invoke, Post-Invoke

This is a list of shell commands to run before/after invoking **dpkg(1)**. Like options this must be specified in list notation. The commands are invoked in order using /bin/sh; should any fail APT will abort.

Pre-Install-Pkgs

This is a list of shell commands to run before invoking **dpkg(1)**. Like options this must be specified in list notation. The commands are invoked in order using /bin/sh; should any fail APT will abort. APT will pass the filenames of all .deb files it is going to install to the commands, one per line on the requested file descriptor, defaulting to standard input.

Version 2 of this protocol sends more information through the requested file descriptor: a line with the text **VERSION 2**, the APT configuration space, and a list of package actions with filename and version information.

Each configuration directive line has the form key=value. Special characters (equal signs, newlines, nonprintable characters, quotation marks, and percent signs in key and newlines, nonprintable characters, and percent signs in value) are %–encoded. Lists are represented by multiple key::=value lines with the same key. The configuration section ends with a blank line.

Package action lines consist of five fields in Version 2: package name (without architecture qualification even if foreign), old version, direction of version change (< for upgrades, > for downgrades, = for no change), new version, action. The version fields are “-” for no version at all (for example when installing a package for the first time; no version is treated as earlier than any real version, so that is an upgrade, indicated as –< 1.23.4). The action field is “**CONFIGURE**” if the package is being configured, “**REMOVE**” if it is being removed, or the filename of a .deb file if it is being unpacked.

In Version 3 after each version field follows the architecture of this version, which is “-” if there is no version, and a field showing the MultiArch type “same”, “foreign”, “allowed” or “none”. Note that “none” is an incorrect typename which is just kept to remain compatible, it should be read as “no” and users are encouraged to support both.

The version of the protocol to be used for the command *cmd* can be chosen by setting DPkg::Tools::options::*cmd*::Version accordingly, the default being version 1. If APT isn't supporting the requested version it will send the information in the highest version it has support for instead.

The file descriptor to be used to send the information can be requested with DPkg::Tools::options::cmd::InfoFD which defaults to 0 for standard input and is available since version 0.9.11. Support for the option can be detected by looking for the environment variable **APT_HOOK_INFO_FD** which contains the number of the used file descriptor as a confirmation.

Run–Directory

APT chdirs to this directory before invoking **dpkg(1)**, the default is `.`

Build–options

These options are passed to **dpkg-buildpackage(1)** when compiling packages; the default is to disable signing and produce all binaries.

DPkg::ConfigurePending

If this option is set APT will call **dpkg --configure --pending** to let **dpkg(1)** handle all required configurations and triggers. This option is activated by default, but deactivating it could be useful if you want to run APT multiple times in a row – e.g. in an installer. In this scenario you could deactivate this option in all but the last run.

PERIODIC AND ARCHIVES OPTIONS

APT::Periodic and APT::Archives groups of options configure behavior of apt periodic updates, which is done by the `/usr/lib/apt/apt.systemd.daily` script. See the top of this script for the brief documentation of these options.

DEBUG OPTIONS

Enabling options in the Debug:: section will cause debugging information to be sent to the standard error stream of the program utilizing the apt libraries, or enable special program modes that are primarily useful for debugging the behavior of apt. Most of these options are not interesting to a normal user, but a few may be:

- Debug::pkgProblemResolver enables output about the decisions made by dist-upgrade, upgrade, install, remove, purge.
- Debug::NoLocking disables all file locking. This can be used to run some operations (for instance, `apt-get -s install`) as a non-root user.
- Debug::pkgDPkgPM prints out the actual command line each time that apt invokes **dpkg(1)**.
- Debug::IdentCdrom disables the inclusion of statfs data in CD-ROM IDs.

A full list of debugging options to apt follows.

Debug::Acquire::cdrom

Print information related to accessing cdrom:// sources.

Debug::Acquire::ftp

Print information related to downloading packages using FTP.

Debug::Acquire::http

Print information related to downloading packages using HTTP.

Debug::Acquire::https

Print information related to downloading packages using HTTPS.

Debug::Acquire::gpgv

Print information related to verifying cryptographic signatures using gpg.

Debug::aptcdrom

Output information about the process of accessing collections of packages stored on CD-ROMs.

Debug::BuildDeps

Describes the process of resolving build-dependencies in **apt-get(8)**.

Debug::Hashes

Output each cryptographic hash that is generated by the apt libraries.

Debug::IdentCDROM

Do not include information from statfs, namely the number of used and free blocks on the CD-ROM filesystem, when generating an ID for a CD-ROM.

Debug::NoLocking

Disable all file locking. For instance, this will allow two instances of “apt-get update” to run at the same time.

Debug::pkgAcquire

Log when items are added to or removed from the global download queue.

Debug::pkgAcquire::Auth

Output status messages and errors related to verifying checksums and cryptographic signatures of downloaded files.

Debug::pkgAcquire::Diffs

Output information about downloading and applying package index list diffs, and errors relating to package index list diffs.

Debug::pkgAcquire::RRed

Output information related to patching apt package lists when downloading index diffs instead of full indices.

Debug::pkgAcquire::Worker

Log all interactions with the sub-processes that actually perform downloads.

Debug::pkgAutoRemove

Log events related to the automatically-installed status of packages and to the removal of unused packages.

Debug::pkgDepCache::AutoInstall

Generate debug messages describing which packages are being automatically installed to resolve dependencies. This corresponds to the initial auto-install pass performed in, e.g., apt-get install, and not to the full apt dependency resolver; see Debug::pkgProblemResolver for that.

Debug::pkgDepCache::Marker

Generate debug messages describing which packages are marked as keep/install/remove while the ProblemResolver does his work. Each addition or deletion may trigger additional actions; they are shown indented two additional spaces under the original entry. The format for each line is MarkKeep, MarkDelete or MarkInstall followed by package-name <a.b.c -> d.e.f | x.y.z> (section) where a.b.c is the current version of the package, d.e.f is the version considered for installation and x.y.z is a newer version, but not considered for installation (because of a low pin score). The later two can be omitted if there is none or if it is the same as the installed version. section is the name of the section the package appears in.

Debug::pkgDPkgPM

When invoking **dpkg**(1), output the precise command line with which it is being invoked, with arguments separated by a single space character.

Debug::pkgDPkgProgressReporting

Output all the data received from **dpkg**(1) on the status file descriptor and any errors encountered while parsing it.

Debug::pkgOrderList

Generate a trace of the algorithm that decides the order in which apt should pass packages to **dpkg**(1).

Debug::pkgPackageManager

Output status messages tracing the steps performed when invoking **dpkg**(1).

Debug::pkgPolicy

Output the priority of each package list on startup.

Debug::pkgProblemResolver

Trace the execution of the dependency resolver (this applies only to what happens when a complex

dependency problem is encountered).

Debug::pkgProblemResolver::ShowScores

Display a list of all installed packages with their calculated score used by the pkgProblemResolver.

The description of the package is the same as described in Debug::pkgDepCache::Marker

Debug::sourceList

Print information about the vendors read from /etc/apt/vendors.list.

Debug::RunScripts

Display the external commands that are called by apt hooks. This includes e.g. the config options DPkg::{Pre,Post}-Invoke or APT::Update::{Pre,Post}-Invoke.

EXAMPLES

/usr/share/doc/apt/examples/configure-index.gz is a configuration file showing example values for all possible options.

FILES

/etc/apt/apt.conf

APT configuration file. Configuration Item: Dir::Etc::Main.

/etc/apt/apt.conf.d/

APT configuration file fragments. Configuration Item: Dir::Etc::Parts.

SEE ALSO

[apt-cache\(8\)](#), [apt-config\(8\)](#), [apt_preferences\(5\)](#).

BUGS

[APT bug page](#)^[1]. If you wish to report a bug in APT, please see /usr/share/doc/debian/bug-reporting.txt or the [reportbug\(1\)](#) command.

AUTHORS

Jason Gunthorpe

APT team

Daniel Burrows <dburrows@debian.org>

Initial documentation of Debug::*.

NOTES

1. APT bug page
<http://bugs.debian.org/src:apt>

NAME

base64 – base64 encode/decode data and print to standard output

SYNOPSIS

base64 [*OPTION*]... [*FILE*]

DESCRIPTION

Base64 encode or decode *FILE*, or standard input, to standard output.

With no *FILE*, or when *FILE* is `-`, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-d, --decode

decode data

-i, --ignore-garbage

when decoding, ignore non-alphabet characters

-w, --wrap=COLS

wrap encoded lines after COLS character (default 76). Use 0 to disable line wrapping

--help display this help and exit

--version

output version information and exit

The data are encoded as described for the base64 alphabet in RFC 4648. When decoding, the input may contain newlines in addition to the bytes of the formal base64 alphabet. Use **--ignore-garbage** to attempt to recover from any other non-alphabet bytes in the encoded stream.

AUTHOR

Written by Simon Josefsson.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/base64>>
or available locally via: info '(coreutils) base64 invocation'

NAME

bash-builtins – bash built-in commands, see **bash(1)**

SYNOPSIS

bash defines the following built-in commands: `:`, `.`, `[`, alias, bg, bind, break, builtin, case, cd, command, compgen, complete, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, fc, fg, getopt, hash, help, history, if, jobs, kill, let, local, logout, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, type, typeset, ulimit, umask, unalias, unset, until, wait, while.

BASH BUILTIN COMMANDS

Unless otherwise noted, each builtin command documented in this section as accepting options preceded by `-` accepts `--` to signify the end of the options. The `:`, **true**, **false**, and **test**/! builtins do not accept options and do not treat `--` specially. The **exit**, **logout**, **r eturn**, **b reak**, **c ontinue**, **l et**, and **s hift** builtins accept and process arguments beginning with `-` without requiring `--`. Other builtins that accept arguments but are not specified as accepting options interpret arguments beginning with `-` as invalid options and require `--` to prevent this interpretation.

: [*arguments*]

No effect; the command does nothing beyond expanding *arguments* and performing any specified redirections. The return status is zero.

. *filename* [*arguments*]

source *filename* [*arguments*]

Read and execute commands from *filename* in the current shell environment and return the exit status of the last command executed from *filename*. If *filename* does not contain a slash, filenames in **PATH** are used to find the directory containing *filename*. The file searched for in **PATH** need not be executable. When **bash** is not in *posix mode*, the current directory is searched if no file is found in **PATH**. If the **sourc epath** option to the **shopt** builtin command is turned off, the **PATH** is not searched. If any *arguments* are supplied, they become the positional parameters when *filename* is executed. Otherwise the positional parameters are unchanged. If the **-T** option is enabled, **source** inherits any trap on **DEBUG**; if it is not, any **DEBUG** trap string is saved and restored around the call to **source**, and **source** unsets the **DEBUG** trap while it executes. If **-T** is not set, and the sourced file changes the **DEBUG** trap, the new value is retained when **source** completes. The return status is the status of the last command exited within the script (0 if no commands are executed), and false if *filename* is not found or cannot be read.

alias [**-p**] [*name*[**=value**] ...]

Alias with no arguments or with the **-p** option prints the list of aliases in the form **alias** *name***=value** on standard output. When arguments are supplied, an alias is defined for each *name* whose *value* is given. A trailing space *invalue* causes the next word to be checked for alias substitution when the alias is expanded. For each *name* in the argument list for which no *value* is supplied, the name and value of the alias is printed. **Alias** returns true unless a *name* is given for which no alias has been defined.

bg [*jobspec* ...]

Resume each suspended job *jobspec* in the background, as if it had been started with **&**. If *jobspec* is not present, the shell's notion of the *current job* is used. **bg** *jobspec* returns 0 unless run when job control is disabled or, when run with job control enabled, any specified *jobspec* was not found or was started without job control.

bind [**-m** *keymap*] [**-lpsvPSVX**]

bind [**-m** *keymap*] [**-q** *function*] [**-u** *function*] [**-r** *keyseq*]

bind [**-m** *keymap*] [**-f** *filename*]

bind [**-m** *keymap*] [**-x** *keyseq:shell-command*]

bind [**-m** *keymap*] [*keyseq:function-name*]

bind [**-m** *keymap*] [*keyseq:readline-command*]

Display current **readline** key and function bindings, bind a key sequence to a **readline** function or macro, or set a **readline** variable. Each non-option argument is a command as it would appear in *.inputrc*, but each binding or command must be passed as a separate argument; e.g., `'"\C-x\C-r"`:

re-read-init-file'. Options, if supplied, have the following meanings:

-m keymap

Use *keymap* as the keymap to be affected by the subsequent bindings. Acceptable *keymap* names are *emacs*, *emacs-standard*, *emacs-meta*, *emacs-ctlx*, *vi*, *vi-move*, *vi-command*, and *vi-insert*. *vi* is equivalent to *vi-command* (*vi-move* is also a synonym); *emacs* is equivalent to *emacs-standard*.

-l List the names of all **readline** functions.

-p Display **readline** function names and bindings in such a way that they can be re-read.

-P List current **readline** function names and bindings.

-s Display **readline** key sequences bound to macros and the strings they output in such a way that they can be re-read.

-S Display **readline** key sequences bound to macros and the strings they output.

-v Display **readline** variable names and values in such a way that they can be re-read.

-V List current **readline** variable names and values.

-f filename

Read key bindings from *filename*.

-q function

Query about which keys invoke the named *function*.

-u function

Unbind all keys bound to the named *function*.

-r keyseq

Remove any current binding for *keyseq*.

-x keyseq:shell-command

Cause *shell-command* to be executed whenever *keyseq* is entered. When *shell-command* is executed, the shell sets the **READLINE_LINE** variable to the contents of the **readline** line buffer and the **READLINE_POINT** and **READLINE_MARK** variables to the current location of the insertion point and the saved insertion point (the mark), respectively. If the executed command changes the value of any of **READLINE_LINE**, **READLINE_POINT**, or **READLINE_MARK**, those new values will be reflected in the editing state.

-X List all key sequences bound to shell commands and the associated commands in a format that can be reused as input.

The return value is 0 unless an unrecognized option is given or an error occurred.

break [n]

Exit from within a **for**, **while**, **until**, or **select** loop. If *n* is specified, break *n* levels. *n* must be ≥ 1 . If *n* is greater than the number of enclosing loops, all enclosing loops are exited. The return value is 0 unless *n* is not greater than or equal to 1.

builtin shell-builtin [arguments]

Execute the specified shell builtin, passing it *arguments*, and return its exit status. This is useful when defining a function whose name is the same as a shell builtin, retaining the functionality of the builtin within the function. The **cd** builtin is commonly redefined this way. The return status is false if *shell-builtin* is not a shell builtin command.

caller [expr]

Returns the context of any active subroutine call (a shell function or a script executed with the . or **source** builtins). Without *expr*, **caller** displays the line number and source filename of the current subroutine call. If a non-negative integer is supplied as *expr*, **caller** displays the line number, subroutine name, and source file corresponding to that position in the current execution call stack. This extra information may be used, for example, to print a stack trace. The current frame is frame 0. The return value is 0 unless the shell is not executing a subroutine call or *expr* does not correspond to a valid position in the call stack.

cd [-L|[-P [-e]] [-@]] [dir]

Change the current directory to *dir*. If *dir* is not supplied, the value of the **HOME** shell variable is the default. Any additional arguments following *dir* are ignored. The variable **CDPATH** defines the search path for the directory containing *dir*: each directory name in **CDPATH** is searched for *dir*. Alternative directory names in **CDPATH** are separated by a colon (:). A null directory name in **CDPATH** is the same as the current directory, i.e., “.”. If *dir* begins with a slash (/), then **CDPATH** is not used. The **-P** option causes **cd** to use the physical directory structure by resolving symbolic links while traversing *dir* and before processing instances of .. in *dir* (see also the **-P** option to the **set** builtin command); the **-L** option forces symbolic links to be followed by resolving the link after processing instances of .. in *dir*. If .. appears in *dir*, it is processed by removing the immediately previous pathname component from *dir*, back to a slash or the beginning of *dir*. If the **-e** option is supplied with **-P**, and the current working directory cannot be successfully determined after a successful directory change, **cd** will return an unsuccessful status. On systems that support it, the **-@** option presents the extended attributes associated with a file as a directory. An argument of – is converted to **\$OLDPWD** before the directory change is attempted. If a non-empty directory name from **CDPATH** is used, or if – is the first argument, and the directory change is successful, the absolute pathname of the new working directory is written to the standard output. The return value is true if the directory was successfully changed; false otherwise.

command [-pVv] command [arg ...]

Run *command* with *args* suppressing the normal shell function lookup. Only builtin commands or commands found in the **PATH** are executed. If the **-p** option is given, the search for *command* is performed using a default value for **PATH** that is guaranteed to find all of the standard utilities. If either the **-V** or **-v** option is supplied, a description of *command* is printed. The **-v** option causes a single word indicating the command or filename used to invoke *command* to be displayed; the **-V** option produces a more verbose description. If the **-V** or **-v** option is supplied, the exit status is 0 if *command* was found, and 1 if not. If neither option is supplied and an error occurred or *command* cannot be found, the exit status is 127. Otherwise, the exit status of the **command** builtin is the exit status of *command*.

compgen [option] [word]

Generate possible completion matches for *word* according to the *options*, which may be any option accepted by the **complete** builtin with the exception of **-p** and **-r**, and write the matches to the standard output. When using the **-F** or **-C** options, the various shell variables set by the programmable completion facilities, while available, will not have useful values.

The matches will be generated in the same way as if the programmable completion code had generated them directly from a completion specification with the same flags. If *word* is specified, only those completions matching *word* will be displayed.

The return value is true unless an invalid option is supplied, or no matches were generated.

complete [-abcdefgjksuv] [-o comp-option] [-DEI] [-A action] [-G globpat] [-W wordlist]

[-F function] [-C command] [-X filterpat] [-P prefix] [-S suffix] name [name ...]

complete -pr [-DEI] [name ...]

Specify how arguments to each *name* should be completed. If the **-p** option is supplied, or if no options are supplied, existing completion specifications are printed in a way that allows them to be reused as input. The **-r** option removes a completion specification for each *name*, or, if no *names* are supplied, all completion specifications. The **-D** option indicates that other supplied options and actions should apply to the “default” command completion; that is, completion attempted on a command for which no completion has previously been defined. The **-E** option indicates that other supplied options and actions should apply to “empty” command completion; that is, completion attempted on a blank line. The **-I** option indicates that other supplied options and actions should apply to completion on the initial non-assignment word on the line, or after a command delimiter such as ; or |, which is usually command name completion. If multiple options are supplied, the **-D** option takes precedence over **-E**, and both take precedence over **-I**. If any of **-D**,

-E, or **-I** are supplied, any other *name* arguments are ignored; these completions only apply to the case specified by the option.

The process of applying these completion specifications when word completion is attempted is described above under **Programmable Completion**.

Other options, if specified, have the following meanings. The arguments to the **-G**, **-W**, and **-X** options (and, if necessary, the **-P** and **-S** options) should be quoted to protect them from expansion before the **complete** builtin is invoked.

-o comp-option

The *comp-option* controls several aspects of the compspec's behavior beyond the simple generation of completions. *comp-option* may be one of:

bashdefault

Perform the rest of the default **bash** completions if the compspec generates no matches.

default Use readline's default filename completion if the compspec generates no matches.

dirnames

Perform directory name completion if the compspec generates no matches.

filenames

Tell readline that the compspec generates filenames, so it can perform any filename-specific processing (like adding a slash to directory names, quoting special characters, or suppressing trailing spaces). Intended to be used with shell functions.

noquote Tell readline not to quote the completed words if they are filenames (quoting filenames is the default).

nosort Tell readline not to sort the list of possible completions alphabetically.

nospace Tell readline not to append a space (the default) to words completed at the end of the line.

plusdirs After any matches defined by the compspec are generated, directory name completion is attempted and any matches are added to the results of the other actions.

-A action

The *action* may be one of the following to generate a list of possible completions:

alias Alias names. May also be specified as **-a**.

arrayvar

Array variable names.

binding Readline key binding names.

builtin Names of shell builtin commands. May also be specified as **-b**.

command

Command names. May also be specified as **-c**.

directory

Directory names. May also be specified as **-d**.

disabled

Names of disabled shell builtins.

enabled Names of enabled shell builtins.

export Names of exported shell variables. May also be specified as **-e**.

file File names. May also be specified as **-f**.

function

Names of shell functions.

group Group names. May also be specified as **-g**.

helptopic

Help topics as accepted by the **help** builtin.

hostname

Hostnames, as taken from the file specified by the **HOSTFILE** shell variable.

job Job names, if job control is active. May also be specified as **-j**.

keyword

Shell reserved words. May also be specified as **-k**.

running Names of running jobs, if job control is active.

service Service names. May also be specified as **-s**.

setopt Valid arguments for the **-o** option to the **set** builtin.

shopt Shell option names as accepted by the **shopt** builtin.

signal Signal names.

stopped Names of stopped jobs, if job control is active.

user User names. May also be specified as **-u**.

variable Names of all shell variables. May also be specified as **-v**.

-C command

command is executed in a subshell environment, and its output is used as the possible completions.

-F function

The shell function *function* is executed in the current shell environment. When the function is executed, the first argument (**\$1**) is the name of the command whose arguments are being completed, the second argument (**\$2**) is the word being completed, and the third argument (**\$3**) is the word preceding the word being completed on the current command line. When it finishes, the possible completions are retrieved from the value of the **COMPREPLY** array variable.

-G globpat

The pathname expansion pattern *globpat* is expanded to generate the possible completions.

-P prefix

prefix is added at the beginning of each possible completion after all other options have been applied.

-S suffix *suffix* is appended to each possible completion after all other options have been applied.**-W wordlist**

The *wordlist* is split using the characters in the **IFS** special variable as delimiters, and each resultant word is expanded. Shell quoting is honored within *wordlist*, in order to provide a mechanism for the words to contain shell metacharacters or characters in the value of **IFS**. The possible completions are the members of the resultant list which match the word being completed.

-X filterpat

filterpat is a pattern as used for pathname expansion. It is applied to the list of possible completions generated by the preceding options and arguments, and each completion matching *filterpat* is removed from the list. A leading **!** in *filterpat* negates the pattern; in this case, any completion not matching *filterpat* is removed.

The return value is true unless an invalid option is supplied, an option other than **-p** or **-r** is supplied without a *name* argument, an attempt is made to remove a completion specification for a *name* for which no specification exists, or an error occurs adding a completion specification.

compopt [-o option] [-DEI] [+o option] [name]

Modify completion options for each *name* according to the *options*, or for the currently-executing completion if no *names* are supplied. If no *options* are given, display the completion options for each *name* or the current completion. The possible values of *option* are those valid for the **complete** builtin described above. The **-D** option indicates that other supplied options should apply to the “default” command completion; that is, completion attempted on a command for which no completion has previously been defined. The **-E** option indicates that other supplied options should apply to “empty” command completion; that is, completion attempted on a blank line. The **-I** option indicates that other supplied options should apply to completion on the initial non-assignment word on the line, or after a command delimiter such as ; or |, which is usually

command name completion.

The return value is true unless an invalid option is supplied, an attempt is made to modify the options for a *name* for which no completion specification exists, or an output error occurs.

continue [*n*]

Resume the next iteration of the enclosing **for**, **while**, **until**, or **select** loop. If *n* is specified, resume at the *n*th enclosing loop. *n* must be ≥ 1 . If *n* is greater than the number of enclosing loops, the last enclosing loop (the “top-level” loop) is resumed. The return value is 0 unless *n* is not greater than or equal to 1.

declare [–aAfGgiIlNrtux] [–p] [*name*[=value] ...]

typeset [–aAfGgiIlNrtux] [–p] [*name*[=value] ...]

Declare variables and/or give them attributes. If *nonames* are given then display the values of variables. The **-p** option will display the attributes and values of each *name*. When **-p** is used with *name* arguments, additional options, other than **-f** and **-F**, are ignored. When **-p** is supplied without *name* arguments, it will display the attributes and values of all variables having the attributes specified by the additional options. If no other options are supplied with **-p**, **declare** will display the attributes and values of all shell variables. The **-f** option will restrict the display to shell functions. The **-F** option inhibits the display of function definitions; only the function name and attributes are printed. If the **extdebug** shell option is enabled using **shopt**, the source file name and line number where each *name* is defined are displayed as well. The **-F** option implies **-f**. The **-g** option forces variables to be created or modified at the global scope, even when **declare** is executed in a shell function. It is ignored in all other cases. The **-I** option causes local variables to inherit the attributes (except the *nameref* attribute) and value of any existing variable with the same *name* at a surrounding scope. If there is no existing variable, the local variable is initially unset. The following options can be used to restrict output to variables with the specified attribute or to give variables attributes:

- a** Each *name* is an indexed array variable (see **Arrays** above).
- A** Each *name* is an associative array variable (see **Arrays** above).
- f** Use function names only.
- i** The variable is treated as an integer; arithmetic evaluation (see **ARITHMETIC EVALUATION** above) is performed when the variable is assigned a value.
- l** When the variable is assigned a value, all upper-case characters are converted to lower-case. The upper-case attribute is disabled.
- n** Give each *name* the *nameref* attribute, making it a name reference to another variable. That other variable is defined by the value of *name*. All references, assignments, and attribute modifications to *name*, except those using or changing the **-n** attribute itself, are performed on the variable referenced by *name*’s value. The *nameref* attribute cannot be applied to array variables.
- r** Make *names* readonly. These names cannot then be assigned values by subsequent assignment statements or unset.
- t** Give each *name* the *trace* attribute. Traced functions inherit the **DEBUG** and **RETURN** traps from the calling shell. The trace attribute has no special meaning for variables.
- u** When the variable is assigned a value, all lower-case characters are converted to upper-case. The lower-case attribute is disabled.
- x** Mark *names* for export to subsequent commands via the environment.

Using ‘+’ instead of ‘-’ turns off the attribute instead, with the exceptions that **+a** and **+A** may not be used to destroy array variables and **+r** will not remove the readonly attribute. When used in a function, **declare** and **typeset** make each *name* local, as with the **local** command, unless the **-g** option is supplied. If a variable name is followed by *=value*, the value of the variable is set to *value*. When using **-a** or **-A** and the compound assignment syntax to create array variables, additional attributes do not take effect until subsequent assignments. The return value is 0 unless an invalid option is encountered, an attempt is made to define a function using **-f** *foo=bar*, an attempt is made to assign a value to a readonly variable, an attempt is made to assign a value to an

array variable without using the compound assignment syntax (see **Arrays** above), one of the *names* is not a valid shell variable name, an attempt is made to turn off readonly status for a readonly variable, an attempt is made to turn off array status for an array variable, or an attempt is made to display a non-existent function with **-f**.

dirs [**-clpv**] [**+n**] [**-n**]

Without options, displays the list of currently remembered directories. The default display is on a single line with directory names separated by spaces. Directories are added to the list with the **pushd** command; the **popd** command removes entries from the list. The current directory is always the first directory in the stack.

- c** Clears the directory stack by deleting all of the entries.
- l** Produces a listing using full pathnames; the default listing format uses a tilde to denote the home directory.
- p** Print the directory stack with one entry per line.
- v** Print the directory stack with one entry per line, prefixing each entry with its index in the stack.
- +n** Displays the *n*th entry counting from the left of the list shown by **dirs** when invoked without options, starting with zero.
- n** Displays the *n*th entry counting from the right of the list shown by **dirs** when invoked without options, starting with zero.

The return value is 0 unless an invalid option is supplied or *n* indexes beyond the end of the directory stack.

disown [**-ar**] [**-h**] [*jobspec* ... | *pid* ...]

Without options, remove each *jobspec* from the table of active jobs. If *jobspec* is not present, and neither the **-a** nor the **-r** option is supplied, the *current job* is used. If the **-h** option is given, each *jobspec* is not removed from the table, but is marked so that **SIGHUP** is not sent to the job if the shell receives a **SIGHUP**. If *nojobspec* is supplied, the **-a** option means to remove or mark all jobs; the **-r** option without a *jobspec* argument restricts operation to running jobs. The return value is 0 unless a *jobspec* does not specify a valid job.

echo [**-neE**] [*arg* ...]

Output the *args*, separated by spaces, followed by a newline. The return status is 0 unless a write error occurs. If **-n** is specified, the trailing newline is suppressed. If the **-e** option is given, interpretation of the following backslash-escaped characters is enabled. The **-E** option disables the interpretation of these escape characters, even on systems where they are interpreted by default. The **xpg_echo** shell option may be used to dynamically determine whether or not **echo** expands these escape characters by default. **echo** does not interpret -- to mean the end of options. **echo** interprets the following escape sequences:

- \a** alert (bell)
- \b** backspace
- \c** suppress further output
- \e**
- \E** an escape character
- \f** form feed
- \n** new line
- \r** carriage return
- \t** horizontal tab
- \v** vertical tab
- ** backslash
- \0nnn** the eight-bit character whose value is the octal value *nnn* (zero to three octal digits)
- \xHH** the eight-bit character whose value is the hexadecimal value *HH* (one or two hex digits)
- \uHHHH** the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHH* (one to four hex digits)

\UHHHHHHHHH

the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHHH-HHH* (one to eight hex digits)

enable [**-a**] [**-dnps**] [**-f** *filename*] [*name* ...]

Enable and disable builtin shell commands. Disabling a builtin allows a disk command which has the same name as a shell builtin to be executed without specifying a full pathname, even though the shell normally searches for builtins before disk commands. If **-n** is used, each *name* is disabled; otherwise, *names* are enabled. For example, to use the **test** binary found via the **PATH** instead of the shell builtin version, run **enable -n test**. The **-f** option means to load the new builtin command *name* from shared object *filename*, on systems that support dynamic loading. The **-d** option will delete a builtin previously loaded with **-f**. If no *name* arguments are given, or if the **-p** option is supplied, a list of shell builtins is printed. With no other option arguments, the list consists of all enabled shell builtins. If **-n** is supplied, only disabled builtins are printed. If **-a** is supplied, the list printed includes all builtins, with an indication of whether or not each is enabled. If **-s** is supplied, the output is restricted to the POSIX *special* builtins. The return value is 0 unless a *name* is not a shell builtin or there is an error loading a new builtin from a shared object.

eval [*arg* ...]

The *args* are read and concatenated together into a single command. This command is then read and executed by the shell, and its exit status is returned as the value of **eval**. If there are no arguments, **eval** returns 0.

exec [**-cl**] [**-a** *name*] [*command* [*arguments*]]

If *command* is specified, it replaces the shell. No new process is created. The *arguments* become the arguments to *command*. If the **-l** option is supplied, the shell places a dash at the beginning of the zeroth argument passed to *command*. This is what *login(1)* does. The **-c** option causes *command* to be executed with an empty environment. If **-a** is supplied, the shell passes *name* as the zeroth argument to the executed command. If *command* cannot be executed for some reason, a non-interactive shell exits, unless the **execfail** shell option is enabled. In that case, it returns failure. An interactive shell returns failure if the file cannot be executed. A subshell exits unconditionally if **exec** fails. If *command* is not specified, any redirections take effect in the current shell, and the return status is 0. If there is a redirection error, the return status is 1.

exit [*n*] Cause the shell to exit with a status of *n*. If *n* is omitted, the exit status is that of the last command executed. A trap on **EXIT** is executed before the shell terminates.

export [**-fn**] [*name*[=*word*]] ...

export -p

The supplied *names* are marked for automatic export to the environment of subsequently executed commands. If the **-f** option is given, the *names* refer to functions. If no *names* are given, or if the **-p** option is supplied, a list of names of all exported variables is printed. The **-n** option causes the export property to be removed from each *name*. If a variable name is followed by **=word**, the value of the variable is set to *word*. **export** returns an exit status of 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **-f** is supplied with a *name* that is not a function.

fc [**-e** *ename*] [**-lnr**] [*first*] [*last*]

fc -s [*pat=rep*] [*cmd*]

The first form selects a range of commands from *first* to *last* from the history list and displays or edits and re-executes them. *First* and *last* may be specified as a string (to locate the last command beginning with that string) or as a number (an index into the history list, where a negative number is used as an offset from the current command number). When listing, a *first* or *last* of 0 is equivalent to **-1** and **-0** is equivalent to the current command (usually the **fc** command); otherwise 0 is equivalent to **-1** and **-0** is invalid. If *last* is not specified, it is set to the current command for listing (so that **fc -1 -10** prints the last 10 commands) and to *first* otherwise. If *first* is not specified, it is set to the previous command for editing and **-16** for listing.

The **-n** option suppresses the command numbers when listing. The **-r** option reverses the order of the commands. If the **-l** option is given, the commands are listed on standard output. Otherwise, the editor given by *ename* is invoked on a file containing those commands. If *ename* is not given, the value of the **FCEDIT** variable is used, and the value of **EDITOR** if **FCEDIT** is not set. If neither variable is set, *vi* is used. When editing is complete, the edited commands are echoed and executed.

In the second form, *command* is re-executed after each instance of *pat* is replaced by *rep*. *Command* is interpreted the same as *first* above. A useful alias to use with this is `r='fc -s'`, so that typing `r cc` runs the last command beginning with *cc* and typing `r` re-executes the last command.

If the first form is used, the return value is 0 unless an invalid option is encountered or *first* or *last* specify history lines out of range. If the **-e** option is supplied, the return value is the value of the last command executed or failure if an error occurs with the temporary file of commands. If the second form is used, the return status is that of the command re-executed, unless *cmd* does not specify a valid history line, in which case **fc** returns failure.

fg [*jobspec*]

Resume *jobspec* in the foreground, and make it the current job. If *jobspec* is not present, the shell's notion of the *current job* is used. The return value is that of the command placed into the foreground, or failure if run when job control is disabled or, when run with job control enabled, if *jobspec* does not specify a valid job or *jobspec* specifies a job that was started without job control.

getopts *optstring name [arg ...]*

getopts is used by shell procedures to parse positional parameters. *optstring* contains the option characters to be recognized; if a character is followed by a colon, the option is expected to have an argument, which should be separated from it by white space. The colon and question mark characters may not be used as option characters. Each time it is invoked, **getopts** places the next option in the shell variable *name*, initializing *name* if it does not exist, and the index of the next argument to be processed into the variable **OPTIND**. **OPTIND** is initialized to 1 each time the shell or a shell script is invoked. When an option requires an argument, **getopts** places that argument into the variable **OPTARG**. The shell does not reset **OPTIND** automatically; it must be manually reset between multiple calls to **getopts** within the same shell invocation if a new set of parameters is to be used.

When the end of options is encountered, **getopts** exits with a return value greater than zero. **OPTIND** is set to the index of the first non-option argument, and *name* is set to `?`.

getopts normally parses the positional parameters, but if more arguments are supplied as *arg* values, **getopts** parses those instead.

getopts can report errors in two ways. If the first character of *optstring* is a colon, *silent* error reporting is used. In normal operation, diagnostic messages are printed when invalid options or missing option arguments are encountered. If the variable **OPTERR** is set to 0, no error messages will be displayed, even if the first character of *optstring* is not a colon.

If an invalid option is seen, **getopts** places `?` into *name* and, if not silent, prints an error message and unsets **OPTARG**. If **getopts** is silent, the option character found is placed in **OPTARG** and no diagnostic message is printed.

If a required argument is not found, and **getopts** is not silent, a question mark (`?`) is placed in *name*, **OPTARG** is unset, and a diagnostic message is printed. If **getopts** is silent, then a colon (`:`) is placed in *name* and **OPTARG** is set to the option character found.

getopts returns true if an option, specified or unspecified, is found. It returns false if the end of

options is encountered or an error occurs.

hash [**-lr**] [**-p** *filename*] [**-dt**] [*name*]

Each time **hash** is invoked, the full pathname of the command *name* is determined by searching the directories in **\$PATH** and remembered. Any previously-remembered pathname is discarded. If the **-p** option is supplied, no path search is performed, and *filename* is used as the full filename of the command. The **-r** option causes the shell to forget all remembered locations. The **-d** option causes the shell to forget the remembered location of each *name*. If the **-t** option is supplied, the full pathname to which each *name* corresponds is printed. If multiple *name* arguments are supplied with **-t**, the *name* is printed before the hashed full pathname. The **-l** option causes output to be displayed in a format that may be reused as input. If no arguments are given, or if only **-l** is supplied, information about remembered commands is printed. The return status is true unless a *name* is not found or an invalid option is supplied.

help [**-dms**] [*pattern*]

Display helpful information about builtin commands. If *pattern* is specified, **help** gives detailed help on all commands matching *pattern*; otherwise help for all the builtins and shell control structures is printed.

- d** Display a short description of each *pattern*
- m** Display the description of each *pattern* in a manpage-like format
- s** Display only a short usage synopsis for each *pattern*

The return status is 0 unless no command matches *pattern*.

history [*n*]

history -c

history -d *offset*

history -d *start-end*

history -anrw [*filename*]

history -p *arg* [*arg* ...]

history -s *arg* [*arg* ...]

With no options, display the command history list with line numbers. Lines listed with a * have been modified. An argument of *n* lists only the last *n* lines. If the shell variable **HISTTIMEFORMAT** is set and not null, it is used as a format string for **strftime(3)** to display the time stamp associated with each displayed history entry. No intervening blank is printed between the formatted time stamp and the history line. If *filename* is supplied, it is used as the name of the history file; if not, the value of **HISTFILE** is used. Options, if supplied, have the following meanings:

- c** Clear the history list by deleting all the entries.

-d *offset*

Delete the history entry at position *offset*. If *offset* is negative, it is interpreted as relative to one greater than the last history position, so negative indices count back from the end of the history, and an index of -1 refers to the current **history -d** command.

-d *start-end*

Delete the history entries between positions *start* and *end*, inclusive. Positive and negative values for *start* and *end* are interpreted as described above.

- a** Append the “new” history lines to the history file. These are history lines entered since the beginning of the current **bash** session, but not already appended to the history file.
- n** Read the history lines not already read from the history file into the current history list. These are lines appended to the history file since the beginning of the current **bash** session.
- r** Read the contents of the history file and append them to the current history list.
- w** Write the current history list to the history file, overwriting the history file’s contents.
- p** Perform history substitution on the following *args* and display the result on the standard output. Does not store the results in the history list. Each *arg* must be quoted to disable normal history expansion.
- s** Store the *args* in the history list as a single entry. The last command in the history list is removed before the *args* are added.

If the **HISTTIMEFORMAT** variable is set, the time stamp information associated with each history entry is written to the history file, marked with the history comment character. When the history file is read, lines beginning with the history comment character followed immediately by a digit are interpreted as timestamps for the following history entry. The return value is 0 unless an invalid option is encountered, an error occurs while reading or writing the history file, an invalid *offset* is supplied as an argument to **-d**, or the history expansion supplied as an argument to **-p** fails.

jobs [**-Inprs**] [*jobspec* ...]

jobs **-x** *command* [*args* ...]

The first form lists the active jobs. The options have the following meanings:

-l List process IDs in addition to the normal information.

-n Display information only about jobs that have changed status since the user was last notified of their status.

-p List only the process ID of the job's process group leader.

-r Display only running jobs.

-s Display only stopped jobs.

If *jobspec* is given, output is restricted to information about that job. The return status is 0 unless an invalid option is encountered or an invalid *jobspec* is supplied.

If the **-x** option is supplied, **jobs** replaces any *jobspec* found in *command* or *args* with the corresponding process group ID, and executes *command* passing it *args*, returning its exit status.

kill [**-s** *sigspec* | **-n** *signum* | **-sigspec**] [*pid* | *jobspec*] ...

kill **-l**|**-L** [*sigspec* | *exit_status*]

Send the signal named by *sigspec* or *signum* to the processes named by *pid* or *jobspec*. *sigspec* is either a case-insensitive signal name such as **SIGKILL** (with or without the **SIG** prefix) or a signal number; *signum* is a signal number. If *sigspec* is not present, then **SIGTERM** is assumed. An argument of **-l** lists the signal names. If any arguments are supplied when **-l** is given, the names of the signals corresponding to the arguments are listed, and the return status is 0. The *exit_status* argument to **-L** is a number specifying either a signal number or the exit status of a process terminated by a signal. The **-L** option is equivalent to **-l**. **kill** returns true if at least one signal was successfully sent, or false if an error occurs or an invalid option is encountered.

let *arg* [*arg* ...]

Each *arg* is an arithmetic expression to be evaluated (see **ARITHMETIC EVALUATION** above). If the last *arg* evaluates to 0, **let** returns 1; 0 is returned otherwise.

local [*option*] [*name*[=*value*] ... | -]

For each argument, a local variable named *name* is created, and assigned *value*. The *option* can be any of the options accepted by **declare**. When **local** is used within a function, it causes the variable *name* to have a visible scope restricted to that function and its children. If *name* is -, the set of shell options is made local to the function in which **local** is invoked: shell options changed using the **set** builtin inside the function are restored to their original values when the function returns. The restore is effected as if a series of **set** commands were executed to restore the values that were in place before the function. With no operands, **local** writes a list of local variables to the standard output. It is an error to use **local** when not within a function. The return status is 0 unless **local** is used outside a function, an invalid *name* is supplied, or *name* is a readonly variable.

logout Exit a login shell.

mapfile [**-d** *delim*] [**-n** *count*] [**-O** *origin*] [**-s** *count*] [**-t**] [**-u** *fd*] [**-C** *callback*] [**-c** *quantum*] [*array*]

readarray [**-d** *delim*] [**-n** *count*] [**-O** *origin*] [**-s** *count*] [**-t**] [**-u** *fd*] [**-C** *callback*] [**-c** *quantum*] [*array*]

Read lines from the standard input into the indexed array variable *array*, or from file descriptor *fd* if the **-u** option is supplied. The variable **MAPFILE** is the default *array*. Options, if supplied, have the following meanings:

-d The first character of *delim* is used to terminate each input line, rather than newline. If *delim* is the empty string, **mapfile** will terminate a line when it reads a NUL character.

- n** Copy at most *count* lines. If *count* is 0, all lines are copied.
- O** Begin assigning to *array* at index *origin*. The default index is 0.
- s** Discard the first *count* lines read.
- t** Remove a trailing *delim* (default newline) from each line read.
- u** Read lines from file descriptor *fd* instead of the standard input.
- C** Evaluate *callback* each time *quantum* lines are read. The **-c** option specifies *quantum*.
- c** Specify the number of lines read between each call to *callback*.

If **-C** is specified without **-c**, the default quantum is 5000. When *callback* is evaluated, it is supplied the index of the next array element to be assigned and the line to be assigned to that element as additional arguments. *callback* is evaluated after the line is read but before the array element is assigned.

If not supplied with an explicit origin, **mapfile** will clear *array* before assigning to it.

mapfile returns successfully unless an invalid option or option argument is supplied, *array* is invalid or unassignable, or if *array* is not an indexed array.

popd [**-n**] [**+n**] [**-n**]

Removes entries from the directory stack. With no arguments, removes the top directory from the stack, and performs a **cd** to the new top directory. Arguments, if supplied, have the following meanings:

- n** Suppresses the normal change of directory when removing directories from the stack, so that only the stack is manipulated.
- +n** Removes the *n*th entry counting from the left of the list shown by **dirs**, starting with zero. For example: **popd +0** removes the first directory, **popd +1** the second.
- n** Removes the *n*th entry counting from the right of the list shown by **dirs**, starting with zero. For example: **popd -0** removes the last directory, **popd -1** the next to last.

If the **popd** command is successful, a **dirs** is performed as well, and the return status is 0. **popd** returns false if an invalid option is encountered, the directory stack is empty, a non-existent directory stack entry is specified, or the directory change fails.

printf [**-v** *var*] *format* [*arguments*]

Write the formatted *arguments* to the standard output under the control of the *format*. The **-v** option causes the output to be assigned to the variable *var* rather than being printed to the standard output.

The *format* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences, which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive *argument*. In addition to the standard **printf(1)** format specifications, **printf** interprets the following extensions:

- %b** causes **printf** to expand backslash escape sequences in the corresponding *argument* in the same way as **echo -e**.
- %q** causes **printf** to output the corresponding *argument* in a format that can be reused as shell input.
- %(*datefmt*)T**

causes **printf** to output the date-time string resulting from using *datefmt* as a format string for **strftime(3)**. The corresponding *argument* is an integer representing the number of seconds since the epoch. Two special argument values may be used: **-1** represents the current time, and **-2** represents the time the shell was invoked. If no argument is specified, conversion behaves as if **-1** had been given. This is an exception to the usual **printf** behavior.

The **%b**, **%q**, and **%T** directives all use the field width and precision arguments from the format specification and write that many bytes from (or use that wide a field for) the expanded argument, which usually contains more characters than the original.

Arguments to non-string format specifiers are treated as C constants, except that a leading plus or minus sign is allowed, and if the leading character is a single or double quote, the value is the ASCII value of the following character.

The *format* is reused as necessary to consume all of the *arguments*. If the *format* requires more *arguments* than are supplied, the extra format specifications behave as if a zero value or null string, as appropriate, had been supplied. The return value is zero on success, non-zero on failure.

pushd [-n] [+n] [-n]

pushd [-n] [dir]

Adds a directory to the top of the directory stack, or rotates the stack, making the new top of the stack the current working directory. With no arguments, **pushd** exchanges the top two directories and returns 0, unless the directory stack is empty. Arguments, if supplied, have the following meanings:

- n** Suppresses the normal change of directory when rotating or adding directories to the stack, so that only the stack is manipulated.
- +n** Rotates the stack so that the *n*th directory (counting from the left of the list shown by **dirs**, starting with zero) is at the top.
- n** Rotates the stack so that the *n*th directory (counting from the right of the list shown by **dirs**, starting with zero) is at the top.
- dir** Adds *dir* to the directory stack at the top, making it the new current working directory as if it had been supplied as the argument to the **cd** builtin.

If the **pushd** command is successful, a **dirs** is performed as well. If the first form is used, **pushd** returns 0 unless the cd to *dir* fails. With the second form, **pushd** returns 0 unless the directory stack is empty, a non-existent directory stack element is specified, or the directory change to the specified new current directory fails.

pwd [-LP]

Print the absolute pathname of the current working directory. The pathname printed contains no symbolic links if the **-P** option is supplied or the **-o physical** option to the **set** builtin command is enabled. If the **-L** option is used, the pathname printed may contain symbolic links. The return status is 0 unless an error occurs while reading the name of the current directory or an invalid option is supplied.

read [-ers] [-a *aname*] [-d *delim*] [-i *text*] [-n *nchars*] [-N *nchars*] [-p *prompt*] [-t *timeout*] [-u *fd*] [*name* ...]

One line is read from the standard input, or from the file descriptor *fd* supplied as an argument to the **-u** option, split into words as described above under **Word Splitting**, and the first word is assigned to the first *name*, the second word to the second *name*, and so on. If there are more words than names, the remaining words and their intervening delimiters are assigned to the last *name*. If there are fewer words read from the input stream than names, the remaining names are assigned empty values. The characters in **IFS** are used to split the line into words using the same rules the shell uses for expansion (described above under **Word Splitting**). The backslash character (\) may be used to remove any special meaning for the next character read and for line continuation. Options, if supplied, have the following meanings:

-a *aname*

The words are assigned to sequential indices of the array variable *aname*, starting at 0. *aname* is unset before any new values are assigned. Other *name* arguments are ignored.

-d *delim*

The first character of *delim* is used to terminate the input line, rather than newline. If *delim* is the empty string, **read** will terminate a line when it reads a NUL character.

-e

If the standard input is coming from a terminal, **readline** (see **READLINE** above) is used to obtain the line. Readline uses the current (or default, if line editing was not previously active) editing settings, but uses Readline's default filename completion.

-i *text*

If **readline** is being used to read the line, *text* is placed into the editing buffer before editing begins.

-n *nchars*

read returns after reading *nchars* characters rather than waiting for a complete line of input, but honors a delimiter if fewer than *nchars* characters are read before the delimiter.

-N *nchars*

read returns after reading exactly *nchars* characters rather than waiting for a complete line of input, unless EOF is encountered or **read** times out. Delimiter characters encountered in the input are not treated specially and do not cause **read** to return until *nchars* characters are read. The result is not split on the characters in **IFS**; the intent is that the variable is assigned exactly the characters read (with the exception of backslash; see the **-r** option below).

-p *prompt*

Display *prompt* on standard error, without a trailing newline, before attempting to read any input. The prompt is displayed only if input is coming from a terminal.

-r

Backslash does not act as an escape character. The backslash is considered to be part of the line. In particular, a backslash-newline pair may not then be used as a line continuation.

-s

Silent mode. If input is coming from a terminal, characters are not echoed.

-t *timeout*

Cause **read** to time out and return failure if a complete line of input (or a specified number of characters) is not read within *timeout* seconds. *timeout* may be a decimal number with a fractional portion following the decimal point. This option is only effective if **read** is reading input from a terminal, pipe, or other special file; it has no effect when reading from regular files. If **read** times out, **read** saves any partial input read into the specified variable *name*. If *timeout* is 0, **read** returns immediately, without trying to read any data. The exit status is 0 if input is available on the specified file descriptor, non-zero otherwise. The exit status is greater than 128 if the timeout is exceeded.

-u *fd*

Read input from file descriptor *fd*.

If no *names* are supplied, the line read, without the ending delimiter but otherwise unmodified, is assigned to the variable **REPLY**. The exit status is zero, unless end-of-file is encountered, **read** times out (in which case the status is greater than 128), a variable assignment error (such as assigning to a readonly variable) occurs, or an invalid file descriptor is supplied as the argument to **-u**.

readonly [-aAf] [-p] [*name[=word]* ...]

The given *names* are marked readonly; the values of these *names* may not be changed by subsequent assignment. If the **-f** option is supplied, the functions corresponding to the *names* are so marked. The **-a** option restricts the variables to indexed arrays; the **-A** option restricts the variables to associative arrays. If both options are supplied, **-A** takes precedence. If no *name* arguments are given, or if the **-p** option is supplied, a list of all readonly names is printed. The other options may be used to restrict the output to a subset of the set of readonly names. The **-p** option causes output to be displayed in a format that may be reused as input. If a variable name is followed by *=word*, the value of the variable is set to *word*. The return status is 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **-f** is supplied with a *name* that is not a function.

return [*n*]

Causes a function to stop executing and return the value specified by *n* to its caller. If *n* is omitted, the return status is that of the last command executed in the function body. If **return** is executed by a trap handler, the last command used to determine the status is the last command executed before the trap handler. If **return** is executed during a **DEBUG** trap, the last command used to determine the status is the last command executed by the trap handler before **return** was invoked. If **return** is used outside a function, but during execution of a script by the **. (source)** command, it causes the shell to stop executing that script and return either *n* or the exit status of the last command executed within the script as the exit status of the script. If *n* is supplied, the return value is its least significant 8 bits. The return status is non-zero if **return** is supplied a non-numeric

argument, or is used outside a function and not during execution of a script by . or **source**. Any command associated with the **RETURN** trap is executed before execution resumes after the function or script.

```
set [--abefhkmnptuvxBCEHPT] [-o option-name] [arg ...]
set [+abefhkmnptuvxBCEHPT] [+o option-name] [arg ...]
```

Without options, the name and value of each shell variable are displayed in a format that can be reused as input for setting or resetting the currently-set variables. Read-only variables cannot be reset. In *Posix mode*, only shell variables are listed. The output is sorted according to the current locale. When options are specified, they set or unset shell attributes. Any arguments remaining after option processing are treated as values for the positional parameters and are assigned, in order, to \$1, \$2, ... \$n. Options, if specified, have the following meanings:

- a** Each variable or function that is created or modified is given the **export** attribute and marked for export to the environment of subsequent commands.
- b** Report the status of terminated background jobs immediately, rather than before the next primary prompt. This is effective only when job control is enabled.
- e** Exit immediately if a *pipeline* (which may consist of a single *simple command*), a *list*, or a *compound command* (see **SHELL GRAMMAR** above), exits with a non-zero status. The shell does not exit if the command that fails is part of the command list immediately following a **while** or **until** keyword, part of the test following the **if** or **elif** reserved words, part of any command executed in a **&&** or **||** list except the command following the final **&&** or **||**, any command in a pipeline but the last, or if the command's return value is being inverted with !. If a compound command other than a subshell returns a non-zero status because a command failed while **-e** was being ignored, the shell does not exit. A trap on **ERR**, if set, is executed before the shell exits. This option applies to the shell environment and each subshell environment separately (see **COMMAND EXECUTION ENVIRONMENT** above), and may cause subshells to exit before executing all the commands in the subshell.

If a compound command or shell function executes in a context where **-e** is being ignored, none of the commands executed within the compound command or function body will be affected by the **-e** setting, even if **-e** is set and a command returns a failure status. If a compound command or shell function sets **-e** while executing in a context where **-e** is ignored, that setting will not have any effect until the compound command or the command containing the function call completes.

- f** Disable pathname expansion.
- h** Remember the location of commands as they are looked up for execution. This is enabled by default.
- k** All arguments in the form of assignment statements are placed in the environment for a command, not just those that precede the command name.
- m** Monitor mode. Job control is enabled. This option is on by default for interactive shells on systems that support it (see **JOB CONTROL** above). All processes run in a separate process group. When a background job completes, the shell prints a line containing its exit status.
- n** Read commands but do not execute them. This may be used to check a shell script for syntax errors. This is ignored by interactive shells.

-o option-name

The *option-name* can be one of the following:

allexport

Same as **-a**.

braceexpand

Same as **-B**.

emacs Use an emacs-style command line editing interface. This is enabled by default when the shell is interactive, unless the shell is started with the **--noediting** option. This also affects the editing interface used for **read -e**.

errexit	Same as -e .
errtrace	Same as -E .
functrace	
	Same as -T .
hashall	Same as -h .
histexpand	
	Same as -H .
history	Enable command history, as described above under HISTORY . This option is on by default in interactive shells.
ignoreeof	The effect is as if the shell command <code>IGNOREEOF=10</code> had been executed (see Shell Variables above).
keyword	
	Same as -k .
monitor	Same as -m .
noclobber	
	Same as -C .
noexec	Same as -n .
noglob	Same as -f .
nolog	Currently ignored.
notify	Same as -b .
nounset	Same as -u .
onecmd	Same as -t .
physical	Same as -P .
pipefail	If set, the return value of a pipeline is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands in the pipeline exit successfully. This option is disabled by default.
posix	Change the behavior of bash where the default operation differs from the POSIX standard to match the standard (<i>posix mode</i>). See SEE ALSO below for a reference to a document that details how posix mode affects bash's behavior.
privileged	
	Same as -p .
verbose	Same as -v .
vi	Use a vi-style command line editing interface. This also affects the editing interface used for read -e .
xtrace	Same as -x .
If -o is supplied with no <i>option-name</i> , the values of the current options are printed. If +o is supplied with no <i>option-name</i> , a series of set commands to recreate the current option settings is displayed on the standard output.	
-p	Turn on <i>privileged</i> mode. In this mode, the \$ENV and \$BASH_ENV files are not processed, shell functions are not inherited from the environment, and the SHELLOPTS , BASHOPTS , CDPATH , and GLOBIGNORE variables, if they appear in the environment, are ignored. If the shell is started with the effective user (group) id not equal to the real user (group) id, and the -p option is not supplied, these actions are taken and the effective user id is set to the real user id. If the -p option is supplied at startup, the effective user id is not reset. Turning this option off causes the effective user and group ids to be set to the real user and group ids.
-t	Exit after reading and executing one command.
-u	Treat unset variables and parameters other than the special parameters "@" and "*" as an error when performing parameter expansion. If expansion is attempted on an unset variable or parameter, the shell prints an error message, and, if not interactive, exits with a non-zero status.

- v** Print shell input lines as they are read.
- x** After expanding each *simple command*, **for** command, **case** command, **select** command, or arithmetic **for** command, display the expanded value of **PS4**, followed by the command and its expanded arguments or associated word list.
- B** The shell performs brace expansion (see **Brace Expansion** above). This is on by default.
- C** If set, **bash** does not overwrite an existing file with the **>**, **>&**, and **<>** redirection operators. This may be overridden when creating output files by using the redirection operator **>|** instead of **>**.
- E** If set, any trap on **ERR** is inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **ERR** trap is normally not inherited in such cases.
- H** Enable ! style history substitution. This option is on by default when the shell is interactive.
- P** If set, the shell does not resolve symbolic links when executing commands such as **cd** that change the current working directory. It uses the physical directory structure instead. By default, **bash** follows the logical chain of directories when performing commands which change the current directory.
- T** If set, any traps on **DEBUG** and **RETURN** are inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **DEBUG** and **RETURN** traps are normally not inherited in such cases.
- If no arguments follow this option, then the positional parameters are unset. Otherwise, the positional parameters are set to the *args*, even if some of them begin with a **-**.
- Signal the end of options, cause all remaining *args* to be assigned to the positional parameters. The **-x** and **-v** options are turned off. If there are no *args*, the positional parameters remain unchanged.

The options are off by default unless otherwise noted. Using + rather than – causes these options to be turned off. The options can also be specified as arguments to an invocation of the shell. The current set of options may be found in **\$-**. The return status is always true unless an invalid option is encountered.

shift [n]

The positional parameters from *n+1* ... are renamed to **\$1** Parameters represented by the numbers **\$#** down to **\$#-n+1** are unset. *n* must be a non-negative number less than or equal to **\$#**. If *n* is 0, no parameters are changed. If *n* is not given, it is assumed to be 1. If *n* is greater than **\$#**, the positional parameters are not changed. The return status is greater than zero if *n* is greater than **\$#** or less than zero; otherwise 0.

shopt [-pqsu] [-o] [optname ...]

Toggle the values of settings controlling optional shell behavior. The settings can be either those listed below, or, if the **-o** option is used, those available with the **-o** option to the **set** builtin command. With no options, or with the **-p** option, a list of all settable options is displayed, with an indication of whether or not each is set; if *optnames* are supplied, the output is restricted to those options. The **-p** option causes output to be displayed in a form that may be reused as input. Other options have the following meanings:

- s** Enable (set) each *optname*.
- u** Disable (unset) each *optname*.
- q** Suppresses normal output (quiet mode); the return status indicates whether the *optname* is set or unset. If multiple *optname* arguments are given with **-q**, the return status is zero if all *optnames* are enabled; non-zero otherwise.
- o** Restricts the values of *optname* to be those defined for the **-o** option to the **set** builtin.

If either **-s** or **-u** is used with no *optname* arguments, **shopt** shows only those options which are set or unset, respectively. Unless otherwise noted, the **shopt** options are disabled (unset) by default.

The return status when listing options is zero if all *optnames* are enabled, non-zero otherwise. When setting or unsetting options, the return status is zero unless an *optname* is not a valid shell option.

The list of **shop** options is:

assoc_expand_once

If set, the shell suppresses multiple evaluation of associative array subscripts during arithmetic expression evaluation, while executing builtins that can perform variable assignments, and while executing builtins that perform array dereferencing.

auto_cd If set, a command name that is the name of a directory is executed as if it were the argument to the **cd** command. This option is only used by interactive shells.

cdable_vars

If set, an argument to the **cd** builtin command that is not a directory is assumed to be the name of a variable whose value is the directory to change to.

cdspell If set, minor errors in the spelling of a directory component in a **cd** command will be corrected. The errors checked for are transposed characters, a missing character, and one character too many. If a correction is found, the corrected filename is printed, and the command proceeds. This option is only used by interactive shells.

checkhash

If set, **bash** checks that a command found in the hash table exists before trying to execute it. If a hashed command no longer exists, a normal path search is performed.

checkjobs

If set, **bash** lists the status of any stopped and running jobs before exiting an interactive shell. If any jobs are running, this causes the exit to be deferred until a second exit is attempted without an intervening command (see **JOB CONTROL** above). The shell always postpones exiting if any jobs are stopped.

checkwinsize

If set, **bash** checks the window size after each external (non-builtin) command and, if necessary, updates the values of **LINES** and **COLUMNS**. This option is enabled by default.

cmdhist If set, **bash** attempts to save all lines of a multiple-line command in the same history entry. This allows easy re-editing of multi-line commands. This option is enabled by default, but only has an effect if command history is enabled, as described above under **HISTORY**.

compat31

compat32

compat40

compat41

compat42

compat43

compat44

These control aspects of the shell's compatibility mode (see **SHELL COMPATIBILITY MODE** below).

complete_fullquote

If set, **bash** quotes all shell metacharacters in filenames and directory names when performing completion. If not set, **bash** removes metacharacters such as the dollar sign from the set of characters that will be quoted in completed filenames when these metacharacters appear in shell variable references in words to be completed. This means that dollar signs in variable names that expand to directories will not be quoted; however, any dollar signs appearing in filenames will not be quoted, either. This is active only when bash is using backslashes to quote completed filenames. This variable is set by default, which is the default bash behavior in versions through 4.2.

direxpand

If set, **bash** replaces directory names with the results of word expansion when performing filename completion. This changes the contents of the readline editing buffer. If not set, **bash** attempts to preserve what the user typed.

dirspell If set, **bash** attempts spelling correction on directory names during word completion if the directory name initially supplied does not exist.

dotglob If set, **bash** includes filenames beginning with a ‘.’ in the results of pathname expansion. The filenames “.” and “..” must always be matched explicitly, even if **dotglob** is set.

execfail If set, a non-interactive shell will not exit if it cannot execute the file specified as an argument to the **exec** builtin command. An interactive shell does not exit if **exec** fails.

expand_aliases

If set, aliases are expanded as described above under **ALIASES**. This option is enabled by default for interactive shells.

extdebug

If set at shell invocation, or in a shell startup file, arrange to execute the debugger profile before the shell starts, identical to the **--debugger** option. If set after invocation, behavior intended for use by debuggers is enabled:

1. The **-F** option to the **declare** builtin displays the source file name and line number corresponding to each function name supplied as an argument.
2. If the command run by the **DEBUG** trap returns a non-zero value, the next command is skipped and not executed.
3. If the command run by the **DEBUG** trap returns a value of 2, and the shell is executing in a subroutine (a shell function or a shell script executed by the **.** or **source** builtins), the shell simulates a call to **return**.
4. **BASH_ARGC** and **BASH_ARGV** are updated as described in their descriptions above.
5. Function tracing is enabled: command substitution, shell functions, and subshells invoked with (*command*) inherit the **DEBUG** and **RETURN** traps.
6. Error tracing is enabled: command substitution, shell functions, and subshells invoked with (*command*) inherit the **ERR** trap.

extglob If set, the extended pattern matching features described above under **Pathname Expansion** are enabled.

extquote

If set, `$'string'` and `$"string"` quoting is performed within `$(parameter)` expansions enclosed in double quotes. This option is enabled by default.

failglob If set, patterns which fail to match filenames during pathname expansion result in an expansion error.

force_ignore

If set, the suffixes specified by the **IGNORE** shell variable cause words to be ignored when performing word completion even if the ignored words are the only possible completions. See **SHELL VARIABLES** above for a description of **IGNORE**. This option is enabled by default.

globasciiranges

If set, range expressions used in pattern matching bracket expressions (see **Pattern Matching** above) behave as if in the traditional C locale when performing comparisons. That is, the current locale's collating sequence is not taken into account, so **b** will not collate between **A** and **B**, and upper-case and lower-case ASCII characters will collate together.

globstar If set, the pattern `**` used in a pathname expansion context will match all files and zero or more directories and subdirectories. If the pattern is followed by a `/`, only directories and subdirectories match.

gnu_errfmt

If set, shell error messages are written in the standard GNU error message format.

histappend

If set, the history list is appended to the file named by the value of the **HISTFILE** variable when the shell exits, rather than overwriting the file.

histreedit

If set, and **readline** is being used, a user is given the opportunity to re-edit a failed history substitution.

histverify

If set, and **readline** is being used, the results of history substitution are not immediately passed to the shell parser. Instead, the resulting line is loaded into the **readline** editing buffer, allowing further modification.

hostcomplete

If set, and **readline** is being used, **bash** will attempt to perform hostname completion when a word containing a `@` is being completed (see **Completing** under **READLINE** above). This is enabled by default.

huponexit

If set, **bash** will send **SIGHUP** to all jobs when an interactive login shell exits.

inherit_errexit

If set, command substitution inherits the value of the **errexit** option, instead of unsetting it in the subshell environment. This option is enabled when *posix mode* is enabled.

interactive_comments

If set, allow a word beginning with `#` to cause that word and all remaining characters on that line to be ignored in an interactive shell (see **COMMENTS** above). This option is enabled by default.

lastpipe If set, and job control is not active, the shell runs the last command of a pipeline not executed in the background in the current shell environment.

lithist If set, and the **cmdhist** option is enabled, multi-line commands are saved to the history with embedded newlines rather than using semicolon separators where possible.

localvar_inherit

If set, local variables inherit the value and attributes of a variable of the same name that exists at a previous scope before any new value is assigned. The **nameref** attribute is not inherited.

localvar_unset

If set, calling **unset** on local variables in previous function scopes marks them so subsequent lookups find them unset until that function returns. This is identical to the behavior of unsetting local variables at the current function scope.

login_shell

The shell sets this option if it is started as a login shell (see **INVOCATION** above). The value may not be changed.

mailwarn

If set, and a file that **bash** is checking for mail has been accessed since the last time it was checked, the message “The mail in *mailfile* has been read” is displayed.

no_empty_cmd_completion

If set, and **readline** is being used, **bash** will not attempt to search the **PATH** for possible completions when completion is attempted on an empty line.

nocaseglob

If set, **bash** matches filenames in a case-insensitive fashion when performing pathname expansion (see **Pathname Expansion** above).

nocasematch

If set, **bash** matches patterns in a case-insensitive fashion when performing matching while executing **case** or **[[** conditional commands, when performing pattern substitution word expansions, or when filtering possible completions as part of programmable completion.

nullglob

If set, **bash** allows patterns which match no files (see **Pathname Expansion** above) to expand to a null string, rather than themselves.

progcomp

If set, the programmable completion facilities (see **Programmable Completion** above) are enabled. This option is enabled by default.

progcomp_alias

If set, and programmable completion is enabled, **bash** treats a command name that doesn't have any completions as a possible alias and attempts alias expansion. If it has an alias, **bash** attempts programmable completion using the command word resulting from the expanded alias.

promptvars

If set, prompt strings undergo parameter expansion, command substitution, arithmetic expansion, and quote removal after being expanded as described in **PROMPTING** above. This option is enabled by default.

restricted_shell

The shell sets this option if it is started in restricted mode (see **RESTRICTED SHELL** below). The value may not be changed. This is not reset when the startup files are executed, allowing the startup files to discover whether or not a shell is restricted.

shift_verbose

If set, the **shift** builtin prints an error message when the shift count exceeds the number of positional parameters.

sourcepath

If set, the **source** (.) builtin uses the value of **PATH** to find the directory containing the file supplied as an argument. This option is enabled by default.

xpg_echo

If set, the **echo** builtin expands backslash-escape sequences by default.

suspend [-f]

Suspend the execution of this shell until it receives a **SIGCONT** signal. A login shell cannot be suspended; the **-f** option can be used to override this and force the suspension. The return status is 0 unless the shell is a login shell and **-f** is not supplied, or if job control is not enabled.

test expr

[*expr*] Return a status of 0 (true) or 1 (false) depending on the evaluation of the conditional expression *expr*. Each operator and operand must be a separate argument. Expressions are composed of the primaries described in the **bash** manual page under **CONDITIONAL EXPRESSIONS**. **test** does not accept any options, nor does it accept and ignore an argument of **--** as signifying the end of options.

Expressions may be combined using the following operators, listed in decreasing order of

precedence. The evaluation depends on the number of arguments; see below. Operator precedence is used when there are five or more arguments.

! expr True if *expr* is false.

(expr) Returns the value of *expr*. This may be used to override the normal precedence of operators.

expr1 -a expr2

True if both *expr1* and *expr2* are true.

expr1 -o expr2

True if either *expr1* or *expr2* is true.

test and **[** evaluate conditional expressions using a set of rules based on the number of arguments.

0 arguments

The expression is false.

1 argument

The expression is true if and only if the argument is not null.

2 arguments

If the first argument is **!**, the expression is true if and only if the second argument is null.

If the first argument is one of the unary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the expression is true if the unary test is true. If the first argument is not a valid unary conditional operator, the expression is false.

3 arguments

The following conditions are applied in the order listed. If the second argument is one of the binary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the result of the expression is the result of the binary test using the first and third arguments as operands. The **-a** and **-o** operators are considered binary operators when there are three arguments. If the first argument is **!**, the value is the negation of the two-argument test using the second and third arguments. If the first argument is exactly **(** and the third argument is exactly **)**, the result is the one-argument test of the second argument. Otherwise, the expression is false.

4 arguments

If the first argument is **!**, the result is the negation of the three-argument expression composed of the remaining arguments. Otherwise, the expression is parsed and evaluated according to precedence using the rules listed above.

5 or more arguments

The expression is parsed and evaluated according to precedence using the rules listed above.

When used with **test** or **[**, the **<** and **>** operators sort lexicographically using ASCII ordering.

times Print the accumulated user and system times for the shell and for processes run from the shell. The return status is 0.

trap [-lp] [[arg] sigspec ...]

The command *arg* is to be read and executed when the shell receives signal(s) *sigspec*. If *arg* is absent (and there is a single *sigspec*) or **-**, each specified signal is reset to its original disposition (the value it had upon entrance to the shell). If *arg* is the null string the signal specified by each *sigspec* is ignored by the shell and by the commands it invokes. If *arg* is not present and **-p** has been supplied, then the trap commands associated with each *sigspec* are displayed. If no arguments are supplied or if only **-p** is given, **trap** prints the list of commands associated with each signal. The **-l** option causes the shell to print a list of signal names and their corresponding numbers. Each *sigspec* is either a signal name defined in *<signal.h>*, or a signal number. Signal names are case insensitive and the **SIG** prefix is optional.

If a *sigspec* is **EXIT (0)** the command *arg* is executed on exit from the shell. If a *sigspec* is **DEBUG**, the command *arg* is executed before every *simple command*, *for* command, *case* command, *select* command, every arithmetic *for* command, and before the first command executes in a shell function (see **SHELL GRAMMAR** above). Refer to the description of the **exitdebug** option to the

shopt builtin for details of its effect on the **DEBUG** trap. If a *sigspec* is **RETURN**, the command *arg* is executed each time a shell function or a script executed with the **.** or **source** builtins finishes executing.

If a *sigspec* is **ERR**, the command *arg* is executed whenever a pipeline (which may consist of a single simple command), a list, or a compound command returns a non-zero exit status, subject to the following conditions. The **ERR** trap is not executed if the failed command is part of the command list immediately following a **while** or **until** keyword, part of the test in an **if** statement, part of a command executed in a **&&** or **||** list except the command following the final **&&** or **||**, any command in a pipeline but the last, or if the command's return value is being inverted using **!**. These are the same conditions obeyed by the **errexit** (**-e**) option.

Signals ignored upon entry to the shell cannot be trapped or reset. Trapped signals that are not being ignored are reset to their original values in a subshell or subshell environment when one is created. The return status is false if any *sigspec* is invalid; otherwise **trap** returns true.

type [**-a****f****t****P**] *name* [*name* ...]

With no options, indicate how each *name* would be interpreted if used as a command name. If the **-t** option is used, **type** prints a string which is one of *alias*, *keyword*, *function*, *builtin*, or *file* if *name* is an alias, shell reserved word, function, builtin, or disk file, respectively. If the *name* is not found, then nothing is printed, and an exit status of false is returned. If the **-p** option is used, **type** either returns the name of the disk file that would be executed if *name* were specified as a command name, or nothing if **type -t** *name* would not return *file*. The **-P** option forces a **P**ATH search for each *name*, even if **type -t** *name* would not return *file*. If a command is hashed, **-p** and **-P** print the hashed value, which is not necessarily the file that appears first in **PATH**. If the **-a** option is used, **type** prints all of the places that contain an executable named *name*. This includes aliases and functions, if and only if the **-p** option is not also used. The table of hashed commands is not consulted when using **-a**. The **-f** option suppresses shell function lookup, as with the **command** builtin. **type** returns true if all of the arguments are found, false if any are not found.

ulimit [**-HS**] **-a**

ulimit [**-HS**] [**-bcdefiklmnpqrstuvxPRT** [*limit*]]

Provides control over the resources available to the shell and to processes started by it, on systems that allow such control. The **-H** and **-S** options specify that the hard or soft limit is set for the given resource. A hard limit cannot be increased by a non-root user once it is set; a soft limit may be increased up to the value of the hard limit. If neither **-H** nor **-S** is specified, both the soft and hard limits are set. The value of *limit* can be a number in the unit specified for the resource or one of the special values **hard**, **soft**, or **unlimited**, which stand for the current hard limit, the current soft limit, and no limit, respectively. If *limit* is omitted, the current value of the soft limit of the resource is printed, unless the **-H** option is given. When more than one resource is specified, the limit name and unit, if appropriate, are printed before the value. Other options are interpreted as follows:

- a** All current limits are reported; no limits are set
- b** The maximum socket buffer size
- c** The maximum size of core files created
- d** The maximum size of a process's data segment
- e** The maximum scheduling priority ("nice")
- f** The maximum size of files written by the shell and its children
- i** The maximum number of pending signals
- k** The maximum number of kqueues that may be allocated
- l** The maximum size that may be locked into memory
- m** The maximum resident set size (many systems do not honor this limit)
- n** The maximum number of open file descriptors (most systems do not allow this value to be set)
- p** The pipe size in 512-byte blocks (this may not be set)

-q	The maximum number of bytes in POSIX message queues
-r	The maximum real-time scheduling priority
-s	The maximum stack size
-t	The maximum amount of cpu time in seconds
-u	The maximum number of processes available to a single user
-v	The maximum amount of virtual memory available to the shell and, on some systems, to its children
-x	The maximum number of file locks
-P	The maximum number of pseudoterminals
-R	The maximum time a real-time process can run before blocking, in microseconds
-T	The maximum number of threads

If *limit* is given, and the **-a** option is not used, *limit* is the new value of the specified resource. If no option is given, then **-f** is assumed. Values are in 1024-byte increments, except for **-t**, which is in seconds; **-R**, which is in microseconds; **-p**, which is in units of 512-byte blocks; **-P**, **-T**, **-b**, **-k**, **-n**, and **-u**, which are unscaled values; and, when in posix mode, **-c** and **-f**, which are in 512-byte increments. The return status is 0 unless an invalid option or argument is supplied, or an error occurs while setting a new limit.

umask [**-p**] [**-S**] [*mode*]

The user file-creation mask is set to *mode*. If *mode* begins with a digit, it is interpreted as an octal number; otherwise it is interpreted as a symbolic mode mask similar to that accepted by *chmod(1)*. If *mode* is omitted, the current value of the mask is printed. The **-S** option causes the mask to be printed in symbolic form; the default output is an octal number. If the **-p** option is supplied, and *mode* is omitted, the output is in a form that may be reused as input. The return status is 0 if the mode was successfully changed or if no *mode* argument was supplied, and false otherwise.

unalias [**-a**] [*name* ...]

Remove each *name* from the list of defined aliases. If **-a** is supplied, all alias definitions are removed. The return value is true unless a supplied *name* is not a defined alias.

unset [**-fv**] [**-n**] [*name* ...]

For each *name*, remove the corresponding variable or function. If the **-v** option is given, each *name* refers to a shell variable, and that variable is removed. Read-only variables may not be unset. If **-f** is specified, each *name* refers to a shell function, and the function definition is removed. If the **-n** option is supplied, and *name* is a variable with the *nameref* attribute, *name* will be unset rather than the variable it references. **-n** has no effect if the **-f** option is supplied. If no options are supplied, each *name* refers to a variable; if there is no variable by that name, a function with that name, if any, is unset. Each unset variable or function is removed from the environment passed to subsequent commands. If any of **BASH_ALIASES**, **BASH_ARGV0**, **BASH_CMDS**, **BASH_COMMAND**, **BASH_SUBSHELL**, **BASHPID**, **COMP_WORDBREAKS**, **DIRSTACK**, **EPOCHREALTIME**, **EPOCHSECONDS**, **FUNCNAME**, **GROUPS**, **HISTCMD**, **LINENO**, **RANDOM**, **SECONDS**, or **SRANDOM** are unset, they lose their special properties, even if they are subsequently reset. The exit status is true unless a *name* is readonly.

wait [**-fn**] [**-p** *varname*] [*id* ...]

Wait for each specified child process and return its termination status. Each *id* may be a process ID or a job specification; if a job spec is given, all processes in that job's pipeline are waited for. If *id* is not given, **wait** waits for all running background jobs and the last-executed process substitution, if its process id is the same as **\$!**, and the return status is zero. If the **-n** option is supplied, **wait** waits for a single job from the list of *ids* or, if no *ids* are supplied, any job, to complete and returns its exit status. If none of the supplied arguments is a child of the shell, or if no arguments are supplied and the shell has no unwaited-for children, the exit status is 127. If the **-p** option is supplied, the process or job identifier of the job for which the exit status is returned is assigned to the variable *varname* named by the option argument. The variable will be unset initially, before any assignment. This is useful only when the **-n** option is supplied. Supplying the **-f** option, when job control is enabled, forces **wait** to wait for *id* to terminate before returning its status,

instead of returning when it changes status. If *id* specifies a non-existent process or job, the return status is 127. Otherwise, the return status is the exit status of the last process or job waited for.

SHELL COMPATIBILITY MODE

Bash-4.0 introduced the concept of a ‘shell compatibility level’, specified as a set of options to the shopt builtin **compat31**, **compat32**, **compat40**, **compat41**, and so on). There is only one current compatibility level -- each option is mutually exclusive. The compatibility level is intended to allow users to select behavior from previous versions that is incompatible with newer versions while they migrate scripts to use current features and behavior. It’s intended to be a temporary solution.

This section does not mention behavior that is standard for a particular version (e.g., setting **compat32** means that quoting the rhs of the regexp matching operator quotes special regexp characters in the word, which is default behavior in bash-3.2 and above).

If a user enables, say, **compat32**, it may affect the behavior of other compatibility levels up to and including the current compatibility level. The idea is that each compatibility level controls behavior that changed in that version of **bash**, but that behavior may have been present in earlier versions. For instance, the change to use locale-based comparisons with the `[[` command came in bash-4.1, and earlier versions used ASCII-based comparisons, so enabling **compat32** will enable ASCII-based comparisons as well. That granularity may not be sufficient for all uses, and as a result users should employ compatibility levels carefully. Read the documentation for a particular feature to find out the current behavior.

Bash-4.3 introduced a new shell variable: **BASH_COMPAT**. The value assigned to this variable (a decimal version number like 4.2, or an integer corresponding to the **compatNN** option, like 42) determines the compatibility level.

Starting with bash-4.4, Bash has begun deprecating older compatibility levels. Eventually, the options will be removed in favor of **BASH_COMPAT**.

Bash-5.0 is the final version for which there will be an individual shopt option for the previous version. Users should use **BASH_COMPAT** on bash-5.0 and later versions.

The following table describes the behavior changes controlled by each compatibility level setting. The **compatNN** tag is used as shorthand for setting the compatibility level to *NN* using one of the following mechanisms. For versions prior to bash-5.0, the compatibility level may be set using the corresponding **compatNN** shopt option. For bash-4.3 and later versions, the **BASH_COMPAT** variable is preferred, and it is required for bash-5.1 and later versions.

compat31

- quoting the rhs of the `[[` command’s regexp matching operator (`=~`) has no special effect

compat32

- interrupting a command list such as "a ; b ; c" causes the execution of the next command in the list (in bash-4.0 and later versions, the shell acts as if it received the interrupt, so interrupting one command in a list aborts the execution of the entire list)

compat40

- the `<` and `>` operators to the `[[` command do not consider the current locale when comparing strings; they use ASCII ordering. Bash versions prior to bash-4.1 use ASCII collation and *strcmp(3)*; bash-4.1 and later use the current locale’s collation sequence and *strcoll(3)*.

compat41

- in *posix* mode, **time** may be followed by options and still be recognized as a reserved word (this is POSIX interpretation 267)
- in *posix* mode, the parser requires that an even number of single quotes occur in the *word* portion of a double-quoted parameter expansion and treats them specially, so that characters within the single quotes are considered quoted (this is POSIX interpretation 221)

compat42

- the replacement string in double-quoted pattern substitution does not undergo quote removal, as it does in versions after bash-4.2

- in posix mode, single quotes are considered special when expanding the *word* portion of a double-quoted parameter expansion and can be used to quote a closing brace or other special character (this is part of POSIX interpretation 221); in later versions, single quotes are not special within double-quoted word expansions

compat43

- the shell does not print a warning message if an attempt is made to use a quoted compound assignment as an argument to declare (declare -a foo='(1 2)'). Later versions warn that this usage is deprecated
- word expansion errors are considered non-fatal errors that cause the current command to fail, even in posix mode (the default behavior is to make them fatal errors that cause the shell to exit)
- when executing a shell function, the loop state (while/until/etc.) is not reset, so **break** or **continue** in that function will break or continue loops in the calling context. Bash-4.4 and later reset the loop state to prevent this

compat44

- the shell sets up the values used by **BASH_ARGV** and **BASH_ARGC** so they can expand to the shell's positional parameters even if extended debugging mode is not enabled
- a subshell inherits loops from its parent context, so **break** or **continue** will cause the subshell to exit. Bash-5.0 and later reset the loop state to prevent the exit
- variable assignments preceding builtins like **export** and **readonly** that set attributes continue to affect variables with the same name in the calling environment even if the shell is not in posix mode

compat50

- Bash-5.1 changed the way **\$RANDOM** is generated to introduce slightly more randomness. If the shell compatibility level is set to 50 or lower, it reverts to the method from bash-5.0 and previous versions, so seeding the random number generator by assigning a value to **RANDOM** will produce the same sequence as in bash-5.0
- If the command hash table is empty, bash versions prior to bash-5.1 printed an informational message to that effect, even when producing output that can be reused as input. Bash-5.1 suppresses that message when the **-I** option is supplied.

SEE ALSO

[bash\(1\)](#), [sh\(1\)](#)

NAME

bash – GNU Bourne-Again SHell

SYNOPSIS

bash [options] [command_string | file]

COPYRIGHT

Bash is Copyright © 1989-2020 by the Free Software Foundation, Inc.

DESCRIPTION

Bash is an **sh**-compatible command language interpreter that executes commands read from the standard input or from a file. **Bash** also incorporates useful features from the *Korn* and *C* shells (**ksh** and **csh**).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). **Bash** can be configured to be POSIX-conformant by default.

OPTIONS

All of the single-character shell options documented in the description of the **set** builtin command, including **-o**, can be used as options when the shell is invoked. In addition, **bash** interprets the following options when it is invoked:

- c** If the **-c** option is present, then commands are read from the first non-option argument *command_string*. If there are arguments after the *command_string*, the first argument is assigned to **\$0** and any remaining arguments are assigned to the positional parameters. The assignment to **\$0** sets the name of the shell, which is used in warning and error messages.
- i** If the **-i** option is present, the shell is *interactive*.
- l** Make **bash** act as if it had been invoked as a login shell (see **INVOCATION** below).
- r** If the **-r** option is present, the shell becomes *restricted* (see **RESTRICTED SHELL** below).
- s** If the **-s** option is present, or if no arguments remain after option processing, then commands are read from the standard input. This option allows the positional parameters to be set when invoking an interactive shell or when reading input through a pipe.
- v** Print shell input lines as they are read.
- x** Print commands and their arguments as they are executed.
- D** A list of all double-quoted strings preceded by **\$** is printed on the standard output. These are the strings that are subject to language translation when the current locale is not **C** or **POSIX**. This implies the **-n** option; no commands will be executed.

[+]-O [shopt_option]

shopt_option is one of the shell options accepted by the **shopt** builtin (see **SHELL BUILTIN COMMANDS** below). If *shopt_option* is present, **-O** sets the value of that option; **+O** unsets it. If *shopt_option* is not supplied, the names and values of the shell options accepted by **shopt** are printed on the standard output. If the invocation option is **+O**, the output is displayed in a format that may be reused as input.

- A **--** signals the end of options and disables further option processing. Any arguments after the **--** are treated as filenames and arguments. An argument of **-** is equivalent to **--**.

Bash also interprets a number of multi-character options. These options must appear on the command line before the single-character options to be recognized.

--debugger

Arrange for the debugger profile to be executed before the shell starts. Turns on extended debugging mode (see the description of the **extdebug** option to the **shopt** builtin below).

--dump-po-strings

Equivalent to **-D**, but the output is in the GNU *gettext po* (portable object) file format.

--dump-strings

Equivalent to **-D**.

--help

Display a usage message on standard output and exit successfully.

--init-file

--rcfile *file*

Execute commands from *file* instead of the system wide initialization file */etc/bash.bashrc* and the standard personal initialization file *~/.bashrc* if the shell is interactive (see **INVOCATION** below).

--login

Equivalent to **-l**.

--noediting

Do not use the GNU **readline** library to read command lines when the shell is interactive.

--noprofile

Do not read either the system-wide startup file */etc/profile* or any of the personal initialization files *~/.bash_profile*, *~/.bash_login*, or *~/.profile*. By default, **bash** reads these files when it is invoked as a login shell (see **INVOCATION** below).

--norc

Do not read and execute the system wide initialization file */etc/bash.bashrc* and the personal initialization file *~/.bashrc* if the shell is interactive. This option is on by default if the shell is invoked as **sh**.

--posix

Change the behavior of **bash** where the default operation differs from the POSIX standard to match the standard (*posix mode*). See **SEE ALSO** below for a reference to a document that details how posix mode affects bash's behavior.

--restricted

The shell becomes restricted (see **RESTRICTED SHELL** below).

--verbose

Equivalent to **-v**.

--version

Show version information for this instance of **bash** on the standard output and exit successfully.

ARGUMENTS

If arguments remain after option processing, and neither the **-c** nor the **-s** option has been supplied, the first argument is assumed to be the name of a file containing shell commands. If **bash** is invoked in this fashion, **\$0** is set to the name of the file, and the positional parameters are set to the remaining arguments. **Bash** reads and executes commands from this file, then exits. **Bash**'s exit status is the exit status of the last command executed in the script. If no commands are executed, the exit status is 0. An attempt is first made to open the file in the current directory, and, if no file is found, then the shell searches the directories in **PATH** for the script.

INVOCATION

A *login shell* is one whose first character of argument zero is a **-**, or one started with the **--login** option.

An *interactive shell* is one started without non-option arguments (unless **-s** is specified) and without the **-c** option whose standard input and error are both connected to terminals (as determined by *isatty(3)*), or one started with the **-i** option. **PS1** is set and **\$-** includes **i** if **bash** is interactive, allowing a shell script or a startup file to test this state.

The following paragraphs describe how **bash** executes its startup files. If any of the files exist but cannot be read, **bash** reports an error. Tildes are expanded in filenames as described below under **Tilde Expansion** in the **EXPANSION** section.

When **bash** is invoked as an interactive login shell, or as a non-interactive shell with the **--login** option, it first reads and executes commands from the file */etc/profile*, if that file exists. After reading that file, it looks for *~/.bash_profile*, *~/.bash_login*, and *~/.profile*, in that order, and reads and executes commands from the first one that exists and is readable. The **--noprofile** option may be used when the shell is started to inhibit this behavior.

When an interactive login shell exits, or a non-interactive login shell executes the **exit** builtin command, **bash** reads and executes commands from the file *~/.bash_logout*, if it exists.

When an interactive shell that is not a login shell is started, **bash** reads and executes commands from */etc/bash.bashrc* and *~/.bashrc*, if these files exist. This may be inhibited by using the **--norc** option. The **--rcfile file** option will force **bash** to read and execute commands from *file* instead of */etc/bash.bashrc* and *~/.bashrc*.

When **bash** is started non-interactively, to run a shell script, for example, it looks for the variable **BASH_ENV** in the environment, expands its value if it appears there, and uses the expanded value as the name of a file to read and execute. **Bash** behaves as if the following command were executed:

```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi
```

but the value of the **PATH** variable is not used to search for the filename.

If **bash** is invoked with the name **sh**, it tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the POSIX standard as well. When invoked as an interactive login shell, or a non-interactive shell with the **--login** option, it first attempts to read and execute commands from */etc/profile* and *~/.profile*, in that order. The **--noprofile** option may be used to inhibit this behavior. When invoked as an interactive shell with the name **sh**, **bash** looks for the variable **ENV**, expands its value if it is defined, and uses the expanded value as the name of a file to read and execute. Since a shell invoked as **sh** does not attempt to read and execute commands from any other startup files, the **--rcfile** option has no effect. A non-interactive shell invoked with the name **sh** does not attempt to read any other startup files. When invoked as **sh**, **bash** enters *posix* mode after the startup files are read.

When **bash** is started in *posix* mode, as with the **--posix** command line option, it follows the POSIX standard for startup files. In this mode, interactive shells expand the **ENV** variable and commands are read and executed from the file whose name is the expanded value. No other startup files are read.

Bash attempts to determine when it is being run with its standard input connected to a network connection, as when executed by the remote shell daemon, usually *rshd*, or the secure shell daemon *sshd*. If **bash** determines it is being run in this fashion, it reads and executes commands from *~/.bashrc* and *~/.bashrc*, if these files exist and are readable. It will not do this if invoked as **sh**. The **--norc** option may be used to inhibit this behavior, and the **--rcfile** option may be used to force another file to be read, but neither *rshd* nor *sshd* generally invoke the shell with those options or allow them to be specified.

If the shell is started with the effective user (group) id not equal to the real user (group) id, and the **-p** option is not supplied, no startup files are read, shell functions are not inherited from the environment, the **SHELLOPTS**, **BASHOPTS**, **CDPATH**, and **GLOBIGNORE** variables, if they appear in the environment, are ignored, and the effective user id is set to the real user id. If the **-p** option is supplied at invocation, the startup behavior is the same, but the effective user id is not reset.

DEFINITIONS

The following definitions are used throughout the rest of this document.

blank A space or tab.

word A sequence of characters considered as a single unit by the shell. Also known as a **token**.

name A *word* consisting only of alphanumeric characters and underscores, and beginning with an alphabetic character or an underscore. Also referred to as an **identifier**.

metacharacter

A character that, when unquoted, separates words. One of the following:

| & ; () < > space tab newline

control operator

A *token* that performs a control function. It is one of the following symbols:

|| & && ; ;; ;& ;;& () | |& <newline>

RESERVED WORDS

Reserved words are words that have a special meaning to the shell. The following words are recognized as reserved when unquoted and either the first word of a command (see **SHELL GRAMMAR** below), the third word of a **case** or **select** command (only **in** is valid), or the third word of a **for** command (only **in** and **do** are valid):

! case coproc do done elif else esac fi for function if in select then

until **while** { } **time** [[]]

SHELL GRAMMAR

Simple Commands

A *simple command* is a sequence of optional variable assignments followed by **blank**-separated words and redirections, and terminated by a *control operator*. The first word specifies the command to be executed, and is passed as argument zero. The remaining words are passed as arguments to the invoked command.

The return value of a *simple command* is its exit status, or 128+*n* if the command is terminated by signal *n*.

Pipelines

A *pipeline* is a sequence of one or more commands separated by one of the control operators | or |&. The format for a pipeline is:

[**time** [-p]] [!] *command* [[| |&] *command2* ...]

The standard output of *command* is connected via a pipe to the standard input of *command2*. This connection is performed before any redirections specified by the command (see **REDIRECTION** below). If |& is used, *command*'s standard error, in addition to its standard output, is connected to *command2*'s standard input through the pipe; it is shorthand for 2>&1|. This implicit redirection of the standard error to the standard output is performed after any redirections specified by the command.

The return status of a pipeline is the exit status of the last command, unless the **pipefail** option is enabled. If **pipefail** is enabled, the pipeline's return status is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands exit successfully. If the reserved word ! precedes a pipeline, the exit status of that pipeline is the logical negation of the exit status as described above. The shell waits for all commands in the pipeline to terminate before returning a value.

If the **time** reserved word precedes a pipeline, the elapsed as well as user and system time consumed by its execution are reported when the pipeline terminates. The -p option changes the output format to that specified by POSIX. When the shell is in *posix mode*, it does not recognize **time** as a reserved word if the next token begins with a '-'. The **TIMEFORMAT** variable may be set to a format string that specifies how the timing information should be displayed; see the description of **TIMEFORMAT** under **Shell Variables** below.

When the shell is in *posix mode*, **time** may be followed by a newline. In this case, the shell displays the total user and system time consumed by the shell and its children. The **TIMEFORMAT** variable may be used to specify the format of the time information.

Each command in a pipeline is executed as a separate process (i.e., in a subshell). See **COMMAND EXECUTION ENVIRONMENT** for a description of a subshell environment. If the **lastpipe** option is enabled using the **shopt** builtin (see the description of **shopt** below), the last element of a pipeline may be run by the shell process.

Lists

A *list* is a sequence of one or more pipelines separated by one of the operators ;, &, &&, or ||, and optionally terminated by one of ;, &, or <newline>.

Of these list operators, && and || have equal precedence, followed by ; and &, which have equal precedence.

A sequence of one or more newlines may appear in a *list* instead of a semicolon to delimit commands.

If a command is terminated by the control operator &, the shell executes the command in the *background* in a subshell. The shell does not wait for the command to finish, and the return status is 0. These are referred to as *asynchronous* commands. Commands separated by a ; are executed sequentially; the shell waits for each command to terminate in turn. The return status is the exit status of the last command executed.

AND and OR lists are sequences of one or more pipelines separated by the && and || control operators, respectively. AND and OR lists are executed with left associativity. An AND list has the form

command1 && *command2*

command2 is executed if, and only if, *command1* returns an exit status of zero (success).

An OR list has the form

command1 || command2

command2 is executed if, and only if, *command1* returns a non-zero exit status. The return status of AND and OR lists is the exit status of the last command executed in the list.

Compound Commands

A *compound command* is one of the following. In most cases *alist* in a command's description may be separated from the rest of the command by one or more newlines, and may be followed by a newline in place of a semicolon.

(*list*) *list* is executed in a subshell environment (see **COMMAND EXECUTION ENVIRONMENT** below). Variable assignments and builtin commands that affect the shell's environment do not remain in effect after the command completes. The return status is the exit status of *list*.

{ *list*; } *list* is simply executed in the current shell environment. *list* must be terminated with a newline or semicolon. This is known as a *group command*. The return status is the exit status of *list*. Note that unlike the metacharacters (and), { and } are *reserved words* and must occur where a reserved word is permitted to be recognized. Since they do not cause a word break, they must be separated from *list* by whitespace or another shell metacharacter.

((*expression*))

The *expression* is evaluated according to the rules described below under **ARITHMETIC EVALUATION**. If the value of the expression is non-zero, the return status is 0; otherwise the return status is 1. This is exactly equivalent to **let "expression"**.

[[*expression*]]

Return a status of 0 or 1 depending on the evaluation of the conditional expression *expression*. Expressions are composed of the primaries described below under **CONDITIONAL EXPRESSIONS**. Word splitting and pathname expansion are not performed on the words between the [[and]]; tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution, and quote removal are performed. Conditional operators such as -f must be unquoted to be recognized as primaries.

When used with [[, the < and > operators sort lexicographically using the current locale.

See the description of the *test* builtin command (section **SHELL BUILTIN COMMANDS** below) for the handling of parameters (i.e. missing parameters).

When the == and != operators are used, the string to the right of the operator is considered a pattern and matched according to the rules described below under **Pattern Matching**, as if the **extglob** shell option were enabled. The = operator is equivalent to ==. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. The return value is 0 if the string matches (==) or does not match (!=) the pattern, and 1 otherwise. Any part of the pattern may be quoted to force the quoted portion to be matched as a string.

An additional binary operator, =~, is available, with the same precedence as == and !=. When it is used, the string to the right of the operator is considered a POSIX extended regular expression and matched accordingly (using the POSIX *regcomp* and *regexec* interfaces usually described in *regex(3)*). The return value is 0 if the string matches the pattern, and 1 otherwise. If the regular expression is syntactically incorrect, the conditional expression's return value is 2. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. Any part of the pattern may be quoted to force the quoted portion to be matched as a string. Bracket expressions in regular expressions must be treated carefully, since normal quoting characters lose their meanings between brackets. If the pattern is stored in a shell variable, quoting the variable expansion forces the entire pattern to be matched as a string.

The pattern will match if it matches any part of the string. Anchor the pattern using the ^ and \$ regular expression operators to force it to match the entire string. The array variable **BASH_REMATCH** records which parts of the string matched the pattern. The element of **BASH_REMATCH** with index 0 contains the portion of the string matching the entire regular expression. Substrings matched by parenthesized

subexpressions within the regular expression are saved in the remaining **BASH_REMATCH** indices. The element of **BASH_REMATCH** with index *n* is the portion of the string matching the *n*th parenthesized subexpression.

Expressions may be combined using the following operators, listed in decreasing order of precedence:

(*expression*)

Returns the value of *expression*. This may be used to override the normal precedence of operators.

! *expression*

True if *expression* is false.

expression1* && *expression2

True if both *expression1* and *expression2* are true.

expression1* || *expression2

True if either *expression1* or *expression2* is true.

The **&&** and **||** operators do not evaluate *expression2* if the value of *expression1* is sufficient to determine the return value of the entire conditional expression.

for *name* [[**in [*word* ...]] ;] **do** *list* ; **done****

The list of words following **in** is expanded, generating a list of items. The variable *name* is set to each element of this list in turn, and *list* is executed each time. If the **in** *word* is omitted, the **for** command executes *list* once for each positional parameter that is set (see **PARAMETERS** below). The return status is the exit status of the last command that executes. If the expansion of the items following **in** results in an empty list, no commands are executed, and the return status is 0.

for ((*expr1* ; *expr2* ; *expr3*)) ; **do *list* ; **done****

First, the arithmetic expression *expr1* is evaluated according to the rules described below under **ARITHMETIC EVALUATION**. The arithmetic expression *expr2* is then evaluated repeatedly until it evaluates to zero. Each time *expr2* evaluates to a non-zero value, *list* is executed and the arithmetic expression *expr3* is evaluated. If any expression is omitted, it behaves as if it evaluates to 1. The return value is the exit status of the last command in *list* that is executed, or false if any of the expressions is invalid.

select *name* [**in *word*] ; **do** *list* ; **done****

The list of words following **in** is expanded, generating a list of items. The set of expanded words is printed on the standard error, each preceded by a number. If the **in** *word* is omitted, the positional parameters are printed (see **PARAMETERS** below). The **PS3** prompt is then displayed and a line read from the standard input. If the line consists of a number corresponding to one of the displayed words, then the value of *name* is set to that word. If the line is empty, the words and prompt are displayed again. If EOF is read, the command completes. Any other value read causes *name* to be set to null. The line read is saved in the variable **REPLY**. The *list* is executed after each selection until a **break** command is executed. The exit status of **select** is the exit status of the last command executed in *list*, or zero if no commands were executed.

case *word* **in [[() *pattern* [| *pattern*] ...) *list* ; ;] ... **esac****

A **case** command first expands *word*, and tries to match it against each *pattern* in turn, using the matching rules described under **Pattern Matching** below. The *word* is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, process substitution and quote removal. Each *pattern* examined is expanded using tilde expansion, parameter and variable expansion, arithmetic expansion, command substitution, and process substitution. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. When a match is found, the corresponding *list* is executed. If the **;;** operator is used, no subsequent matches are attempted after the first pattern match. Using **; &** in place of **;;** causes execution to continue with the *list* associated with the next set of patterns. Using **;; &** in place of **;;** causes the shell to test the next pattern list in the statement, if any, and execute any associated *list* on a successful match, continuing the case statement execution as if the pattern list had not matched. The exit status is zero if no pattern matches. Otherwise, it is the exit status of

the last command executed in *list*.

if *list*; **then** *list*; [**elif** *list*; **then** *list*;] ... [**else** *list*;] **fi**

The **if** *list* is executed. If its exit status is zero, the **then** *list* is executed. Otherwise, each **elif** *list* is executed in turn, and if its exit status is zero, the corresponding **then** *list* is executed and the command completes. Otherwise, the **else** *list* is executed, if present. The exit status is the exit status of the last command executed, or zero if no condition tested true.

while *list-1*; **do** *list-2*; **done**

until *list-1*; **do** *list-2*; **done**

The **while** command continuously executes the list *list-2* as long as the last command in the list *list-1* returns an exit status of zero. The **until** command is identical to the **while** command, except that the test is negated: *list-2* is executed as long as the last command in *list-1* returns a non-zero exit status. The exit status of the **while** and **until** commands is the exit status of the last command executed in *list-2*, or zero if none was executed.

Coprocesses

A *coprocess* is a shell command preceded by the **coproc** reserved word. A coprocess is executed asynchronously in a subshell, as if the command had been terminated with the & control operator, with a two-way pipe established between the executing shell and the coprocess.

The format for a coprocess is:

coproc [*NAME*] *command* [*redirections*]

This creates a coprocess named *NAME*. If *NAME* is not supplied, the default name is **COPROC**. *NAME* must not be supplied if *command* is a *simple command* (see above); otherwise, it is interpreted as the first word of the simple command. When the coprocess is executed, the shell creates an array variable (see **Arrays** below) named *NAME* in the context of the executing shell. The standard output of *command* is connected via a pipe to a file descriptor in the executing shell, and that file descriptor is assigned to *NAME*[0]. The standard input of *command* is connected via a pipe to a file descriptor in the executing shell, and that file descriptor is assigned to *NAME*[1]. This pipe is established before any redirections specified by the command (see **REDIRECTION** below). The file descriptors can be utilized as arguments to shell commands and redirections using standard word expansions. Other than those created to execute command and process substitutions, the file descriptors are not available in subshells. The process ID of the shell spawned to execute the coprocess is available as the value of the variable *NAME_PID*. The **wait** builtin command may be used to wait for the coprocess to terminate.

Since the coprocess is created as an asynchronous command, the **coproc** command always returns success. The return status of a coprocess is the exit status of *command*.

Shell Function Definitions

A shell function is an object that is called like a simple command and executes a compound command with a new set of positional parameters. Shell functions are declared as follows:

fname () *compound-command* [*redirection*]

function *fname* [() *compound-command* [*redirection*]]

This defines a function named *fname*. The reserved word **function** is optional. If the **function** reserved word is supplied, the parentheses are optional. The *body* of the function is the compound command *compound-command* (see **Compound Commands** above). That command is usually a *list* of commands between { and }, but may be any command listed under **Compound Commands** above, with one exception: If the **function** reserved word is used, but the parentheses are not supplied, the braces are required. *compound-command* is executed whenever *fname* is specified as the name of a simple command. When in *posix mode*, *fname* must be a valid shell *name* and may not be the name of one of the *POSIX special builtins*. In default mode, a function name can be any unquoted shell word that does not contain \$. Any redirections (see **REDIRECTION** below) specified when a function is defined are performed when the function is executed. The exit status of a function definition is zero unless a syntax error occurs or a readonly function with the same name already exists. When executed, the exit status of a function is the exit status of the last command executed in the body. (See **FUNCTIONS** below.)

COMMENTS

In a non-interactive shell, or an interactive shell in which the **interactive_comments** option to the **shopt** builtin is enabled (see **SHELL BUILTIN COMMANDS** below), a word beginning with # causes that word and all remaining characters on that line to be ignored. An interactive shell without the **interactive_comments** option enabled does not allow comments. The **interactive_comments** option is on by default in interactive shells.

QUOTING

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to disable special treatment for special characters, to prevent reserved words from being recognized as such, and to prevent parameter expansion.

Each of the *metacharacters* listed above under **DEFINITIONS** has special meaning to the shell and must be quoted if it is to represent itself.

When the command history expansion facilities are being used (see **HISTORY EXPANSION** below), the *history expansion* character, usually !, must be quoted to prevent history expansion.

There are three quoting mechanisms: the *escape character*, single quotes, and double quotes.

A non-quoted backslash () is the *escape character*. It preserves the literal value of the next character that follows, with the exception of <newline>. If a\<ne wline> pair appears, and the backslash is not itself quoted, the \<newline> is treated as a line continuation (that is, it is removed from the input stream and effectively ignored).

Enclosing characters in single quotes preserves the literal value of each character within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash.

Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of \$, ` , \, and, when history expansion is enabled, !. When the shell is *posix mode*, the ! has no special meaning within double quotes, even when history expansion is enabled. The characters \$ and ` retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: \$, ` , " , \, or <newline>. A double quote may be quoted within double quotes by preceding it with a backslash. If enabled, history expansion will be performed unless an ! appearing in double quotes is escaped using a backslash. The backslash preceding the ! is not removed.

The special parameters * and @ have special meaning when in double quotes (see **PARAMETERS** below).

Words of the form \$'string' are treated specially. The word expands to *string*, with backslash-escaped characters replaced as specified by the ANSI C standard. Backslash escape sequences, if present, are decoded as follows:

\a	alert (bell)
\b	backspace
\e	
\E	an escape character
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\v	vertical tab
\\\	backslash
\'	single quote
\\"	double quote
\?	question mark
\nnn	the eight-bit character whose value is the octal value <i>nnn</i> (one to three octal digits)
\xHH	the eight-bit character whose value is the hexadecimal value <i>HH</i> (one or two hex digits)
\uHHHH	the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value <i>HHHH</i> (one to four hex digits)

`\UHHHHHHHHH`

the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHHHHH*
HHH (one to eight hex digits)

`\cx` a control-x character

The expanded result is single-quoted, as if the dollar sign had not been present.

A double-quoted string preceded by a dollar sign (`$$string`) will cause the string to be translated according to the current locale. The `gettext` infrastructure performs the message catalog lookup and translation, using the **LC_MESSAGES** and **TEXTDOMAIN** shell variables. If the current locale isC or **POSIX**, or if there are no translations available, the dollar sign is ignored. If the string is translated and replaced, the replacement is double-quoted.

PARAMETERS

A *parameter* is an entity that stores values. It can be an *ame*, a number , or one of the special characters listed below under **Special Parameters**. A *variable* is a parameter denoted by a *ame*. A *v* *ariable* has a *value* and zero or more *tributes*. Attributes are assigned using the **declare** builtin command (see **declare** below in **SHELL BUILTIN COMMANDS**).

A parameter is set if it has been assigned a value. The null string is a valid value. Once a variable is set, it may be unset only by using the **unset** builtin command (see **SHELL BUILTIN COMMANDS** below).

A *variable* may be assigned to by a statement of the form

`name=[value]`

If *value* is not given, the variable is assigned the null string. All *values* undergo tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal (see **EXPANSION** below). If the variable has its **integer** attribute set, *value* is evaluated as an arithmetic expression even if the `$((...))` expansion is not used (see **Arithmetic Expansion** below). Word splitting is not performed, with the exception of `"$@"` as explained below under **Special Parameters**. Pathname expansion is not performed. Assignment statements may also appear as arguments to the **alias**, **declare**, **typeset**, **export**, **readonly**, and **local** builtin commands (*declaration commands*). When in *posix mode*, these builtins may appear in a command after one or more instances of the **command** builtin and retain these assignment statement properties.

In the context where an assignment statement is assigning a value to a shell variable or array index, the `+=` operator can be used to append to or add to the variable's previous value. This includes arguments to builtin commands such as **declare** that accept assignment statements (*declaration commands*). When `+=` is applied to a variable for which the **integer** attribute has been set, *value* is evaluated as an arithmetic expression and added to the variable's current value, which is also evaluated. When `+=` is applied to an array variable using compound assignment (see **Arrays** below), the variable's value is not unset (as it is when using `=`), and new values are appended to the array beginning at one greater than the array's maximum index (for indexed arrays) or added as additional key-value pairs in an associative array. When applied to a string-valued variable, *value* is expanded and appended to the variable's value.

A variable can be assigned the *nameref* attribute using the `-n` option to the **declare** or **local** builtin commands (see the descriptions of **declare** and **local** below) to create a *nameref*, or a reference to another variable. This allows variables to be manipulated indirectly. Whenever the *nameref* variable is referenced, assigned to, unset, or has its attributes modified (other than using or changing the *nameref* attribute itself), the operation is actually performed on the variable specified by the *nameref* variable's value. A *nameref* is commonly used within shell functions to refer to a variable whose name is passed as an argument to the function. For instance, if a variable name is passed to a shell function as its first argument, running

```
declare -n ref=$1
```

inside the function creates a *nameref* variable **ref** whose value is the variable name passed as the first argument. References and assignments to **ref**, and changes to its attributes, are treated as references, assignments, and attribute modifications to the variable whose name was passed as `$1`. If the control variable in a **for** loop has the *nameref* attribute, the list of words can be a list of shell variables, and a name reference will be established for each word in the list, in turn, when the loop is executed. Array variables cannot be

given the **nameref** attribute. However, nameref variables can reference array variables and subscripted array variables. Namerefs can be unset using the **-n** option to the **unset** builtin. Otherwise, if **unset** is executed with the name of a nameref variable as an argument, the variable referenced by the nameref variable will be unset.

Positional Parameters

A *positional parameter* is a parameter denoted by one or more digits, other than the single digit 0. Positional parameters are assigned from the shell's arguments when it is invoked, and may be reassigned using the **set** builtin command. Positional parameters may not be assigned to with assignment statements. The positional parameters are temporarily replaced when a shell function is executed (see **FUNCTIONS** below).

When a positional parameter consisting of more than a single digit is expanded, it must be enclosed in braces (see **EXPANSION** below).

Special Parameters

The shell treats several parameters specially. These parameters may only be referenced; assignment to them is not allowed.

- * Expands to the positional parameters, starting from one. When the expansion is not within double quotes, each positional parameter expands to a separate word. In contexts where it is performed, those words are subject to further word splitting and pathname expansion. When the expansion occurs within double quotes, it expands to a single word with the value of each parameter separated by the first character of the **IFS** special variable. That is, "\$*" is equivalent to "\$1\$c\$2c...", where *c* is the first character of the value of the **IFS** variable. If **IFS** is unset, the parameters are separated by spaces. If **IFS** is null, the parameters are joined without intervening separators.
- @ Expands to the positional parameters, starting from one. In contexts where word splitting is performed, this expands each positional parameter to a separate word; if not within double quotes, these words are subject to word splitting. In contexts where word splitting is not performed, this expands to a single word with each positional parameter separated by a space. When the expansion occurs within double quotes, each parameter expands to a separate word. That is, "\$@" is equivalent to "\$1" "\$2" ... If the double-quoted expansion occurs within a word, the expansion of the first parameter is joined with the beginning part of the original word, and the expansion of the last parameter is joined with the last part of the original word. When there are no positional parameters, "\$@" and \$@ expand to nothing (i.e., they are removed).
- # Expands to the number of positional parameters in decimal.
- ? Expands to the exit status of the most recently executed foreground pipeline.
- Expands to the current option flags as specified upon invocation, by the **set** builtin command, or those set by the shell itself (such as the **-i** option).
- \$ Expands to the process ID of the shell. In a () subshell, it expands to the process ID of the current shell, not the subshell.
- ! Expands to the process ID of the job most recently placed into the background, whether executed as an asynchronous command or using the **bg** builtin (see **JOB CONTROL** below).
- 0 Expands to the name of the shell or shell script. This is set at shell initialization. If **bash** is invoked with a file of commands, \$0 is set to the name of that file. If **bash** is started with the **-c** option, then \$0 is set to the first argument after the string to be executed, if one is present. Otherwise, it is set to the filename used to invoke **bash**, as given by argument zero.

Shell Variables

The following variables are set by the shell:

- At shell startup, set to the pathname used to invoke the shell or shell script being executed as passed in the environment or argument list. Subsequently, expands to the last argument to the previous simple command executed in the foreground, after expansion. Also set to the full pathname used to invoke each command executed and placed in the environment exported to that command. When checking mail, this parameter holds the name of the mail file currently being checked.
- BASH** Expands to the full filename used to invoke this instance of **bash**.

BASHOPTS

A colon-separated list of enabled shell options. Each word in the list is a valid argument for the `-s` option to the `shopt` builtin command (see **SHELL BUILTIN COMMANDS** below). The options appearing in **BASHOPTS** are those reported as *on* by `shopt`. If this variable is in the environment when **bash** starts up, each shell option in the list will be enabled before reading any startup files. This variable is read-only.

BASHPID

Expands to the process ID of the current **bash** process. This differs from `$$` under certain circumstances, such as subshells that do not require **bash** to be re-initialized. Assignments to **BASHPID** have no effect. If **ASHPID** is unset, it loses its special properties, even if it is subsequently reset.

BASH_ALIASES

An associative array variable whose members correspond to the internal list of aliases as maintained by the `alias` builtin. Elements added to this array appear in the alias list; however, unsetting array elements currently does not cause aliases to be removed from the alias list. If **BASH_ALIASES** is unset, it loses its special properties, even if it is subsequently reset.

BASH_ARGC

An array variable whose values are the number of parameters in each frame of the current **bash** execution call stack. The number of parameters to the current subroutine (shell function or script executed with `.` or `source`) is at the top of the stack. When a subroutine is executed, the number of parameters passed is pushed onto **BASH_ARGC**. The shell sets **BASH_ARGC** only when in extended debugging mode (see the description of the `extdebug` option to the `shopt` builtin below). Setting `extdebug` after the shell has started to execute a script, or referencing this variable when `extdebug` is not set, may result in inconsistent values.

BASH_ARGV

An array variable containing all of the parameters in the current **bash** execution call stack. The final parameter of the last subroutine call is at the top of the stack; the first parameter of the initial call is at the bottom. When a subroutine is executed, the parameters supplied are pushed onto **BASH_ARGV**. The shell sets **BASH_ARGV** only when in extended debugging mode (see the description of the `extdebug` option to the `shopt` builtin below). Setting `extdeb ug` after the shell has started to execute a script, or referencing this variable when `extdebug` is not set, may result in inconsistent values.

BASH_ARGV0

When referenced, this variable expands to the name of the shell or shell script (identical to `$0`; see the description of special parameter 0 above). Assignment to **ASH_ARGV0** causes the value assigned to also be assigned to `$0`. If **ASH_ARGV0** is unset, it loses its special properties, even if it is subsequently reset.

BASH_CMDS

An associative array variable whose members correspond to the internal hash table of commands as maintained by the `hash` builtin. Elements added to this array appear in the hash table; however, unsetting array elements currently does not cause command names to be removed from the hash table. If **ASH_CMDS** is unset, it loses its special properties, even if it is subsequently reset.

BASH_COMMAND

The command currently being executed or about to be executed, unless the shell is executing a command as the result of a trap, in which case it is the command executing at the time of the trap.

If **BASH_COMMAND** is unset, it loses its special properties, even if it is subsequently reset.

BASH_EXECUTION_STRING

The command argument to the `-c` invocation option.

BASH_LINENO

An array variable whose members are the line numbers in source files where each corresponding member of `FUNCNAME` was invoked. `$(BASH_LINENO[${$i}])` is the line number in the source file `$(BASH_SOURCE[${$i+1}])` where `$(FUNCNAME[${$i}])` was called (or `$(BASH_LINENO[${$i-1}])` if referenced within another shell function). Use `LINENO` to obtain the current line number.

BASH_LOADABLES_PATH

A colon-separated list of directories in which the shell looks for dynamically loadable builtins specified by the **enable** command.

BASH_REMATCH

An array variable whose members are assigned by the `=~` binary operator to the `[[` conditional command. The element with index 0 is the portion of the string matching the entire regular expression. The element with index *n* is the portion of the string matching the *n*th parenthesized subexpression.

BASH_SOURCE

An array variable whose members are the source filenames where the corresponding shell function names in the **FUNCNAME** array variable are defined. The shell function `${FUNCNAME[$i]}` is defined in the file `${BASH_SOURCE[$i]}` and called from `${BASH_SOURCE[$i+1]}` .

BASH_SUBSHELL

Incremented by one within each subshell or subshell environment when the shell begins executing in that environment. The initial value is 0. If **BASH_SUBSHELL** is unset, it loses its special properties, even if it is subsequently reset.

BASH_VERSINFO

A readonly array variable whose members hold version information for this instance of **bash**. The values assigned to the array members are as follows:

BASH_VERSINFO[0]	The major version number (the <i>release</i>).
BASH_VERSINFO[1]	The minor version number (the <i>version</i>).
BASH_VERSINFO[2]	The patch level.
BASH_VERSINFO[3]	The build version.
BASH_VERSINFO[4]	The release status (e.g., <i>beta</i>).
BASH_VERSINFO[5]	The value of MACHTYPE .

BASH_VERSION

Expands to a string describing the version of this instance of **bash**.

COMP_CWORD

An index into `${COMP_WORDS}` of the word containing the current cursor position. This variable is available only in shell functions invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_KEY

The key (or final key of a key sequence) used to invoke the current completion function.

COMP_LINE

The current command line. This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_POINT

The index of the current cursor position relative to the beginning of the current command. If the current cursor position is at the end of the current command, the value of this variable is equal to `${#COMP_LINE}` . This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_TYPE

Set to an integer value corresponding to the type of completion attempted that caused a completion function to be called: `TAB`, for normal completion, `?`, for listing completions after successive tabs, `!`, for listing alternatives on partial word completion, `@`, to list completions if the word is not unmodified, or `%`, for menu completion. This variable is available only in shell functions and external commands invoked by the programmable completion facilities (see **Programmable Completion** below).

COMP_WORDBREAKS

The set of characters that the **readline** library treats as word separators when performing word completion. If **COMP_WORDBREAKS** is unset, it loses its special properties, even if it is subsequently reset.

COMP_WORDS

An array variable (see **Arrays** below) consisting of the individual words in the current command line. The line is split into words as **readline** would split it, using **COMP_WORDBREAKS** as described above. This variable is available only in shell functions invoked by the programmable completion facilities (see **Programmable Completion** below).

COPROC

An array variable (see **Arrays** below) created to hold the file descriptors for output from and input to an unnamed coprocess (see **Coprocesses** above).

DIRSTACK

An array variable (see **Arrays** below) containing the current contents of the directory stack. Directories appear in the stack in the order they are displayed by the **dirs** builtin. Assigning to members of this array variable may be used to modify directories already in the stack, but the **pushd** and **popd** builtins must be used to add and remove directories. Assignment to this variable will not change the current directory. If **DIRSTACK** is unset, it loses its special properties, even if it is subsequently reset.

EPOCHREALTIME

Each time this parameter is referenced, it expands to the number of seconds since the Unix Epoch (see *time(3)*) as a floating point value with micro-second granularity. Assignments to **EPOCHREALTIME** are ignored. If **EPOCHREALTIME** is unset, it loses its special properties, even if it is subsequently reset.

EPOCHSECONDS

Each time this parameter is referenced, it expands to the number of seconds since the Unix Epoch (see *time(3)*). Assignments to **EPOCHSECONDS** are ignored. If **EPOCHSECONDS** is unset, it loses its special properties, even if it is subsequently reset.

EUID Expands to the effective user ID of the current user, initialized at shell startup. This variable is readonly.

FUNCNAME

An array variable containing the names of all shell functions currently in the execution call stack. The element with index 0 is the name of any currently-executing shell function. The bottom-most element (the one with the highest index) is "main". This variable exists only when a shell function is executing. Assignments to **FUNCNAME** have no effect. If **FUNCNAME** is unset, it loses its special properties, even if it is subsequently reset.

This variable can be used with **BASH_LINENO** and **BASH_SOURCE**. Each element of **FUNCNAME** has corresponding elements in **BASH_LINENO** and **BASH_SOURCE** to describe the call stack. For instance, **FUNCNAME[*i*]** was called from the file **BASH_SOURCE[*i*+1]** at line number **BASH_LINENO[*i*]**. The **caller** builtin displays the current call stack using this information.

GROUPS

An array variable containing the list of groups of which the current user is a member. Assignments to **GROUPS** have no effect. If **GROUPS** is unset, it loses its special properties, even if it is subsequently reset.

HISTCMD

The history number, or index in the history list, of the current command. Assignments to **HISTCMD** are ignored. If **HISTCMD** is unset, it loses its special properties, even if it is subsequently reset.

HOSTNAME

Automatically set to the name of the current host.

HOSTTYPE

Automatically set to a string that uniquely describes the type of machine on which **bash** is executing. The default is system-dependent.

LINENO

Each time this parameter is referenced, the shell substitutes a decimal number representing the current sequential line number (starting with 1) within a script or function. When not in a script or function, the value substituted is not guaranteed to be meaningful. If **LINENO** is unset, it loses its

special properties, even if it is subsequently reset.

MACHTYPE

Automatically set to a string that fully describes the system type on which **bash** is executing, in the standard GNU *cpu-company-system* format. The default is system-dependent.

MAPFILE

An array variable (see **Arrays** below) created to hold the text read by the **mapfile** builtin when no variable name is supplied.

OLDPWD

The previous working directory as set by the **cd** command.

OPTARG

The value of the last option argument processed by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below).

OPTIND

The index of the next argument to be processed by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below).

OSTYPE

Automatically set to a string that describes the operating system on which **bash** is executing. The default is system-dependent.

PIPESTATUS

An array variable (see **Arrays** below) containing a list of exit status values from the processes in the most-recently-executed foreground pipeline (which may contain only a single command).

PPID

The process ID of the shell's parent. This variable is readonly.

PWD

The current working directory as set by the **cd** command.

RANDOM

Each time this parameter is referenced, it expands to a random integer between 0 and 32767. Assigning a value to **RANDOM** initializes (seeds) the sequence of random numbers. If **RANDOM** is unset, it loses its special properties, even if it is subsequently reset.

READLINE_LINE

The contents of the **readline** line buffer, for use with **bind -x** (see **SHELL BUILTIN COMMANDS** below).

READLINE_MARK

The position of the mark (saved insertion point) in the **readline** line buffer, for use with **bind -x** (see **SHELL BUILTIN COMMANDS** below). The characters between the insertion point and the mark are often called the *region*.

READLINE_POINT

The position of the insertion point in the **readline** line buffer, for use with **bind -x** (see **SHELL BUILTIN COMMANDS** below).

REPLY

Set to the line of input read by the **read** builtin command when no arguments are supplied.

SECONDS

Each time this parameter is referenced, the number of seconds since shell invocation is returned. If a value is assigned to **SECONDS**, the value returned upon subsequent references is the number of seconds since the assignment plus the value assigned. The number of seconds at shell invocation and the current time is always determined by querying the system clock. If **SECONDS** is unset, it loses its special properties, even if it is subsequently reset.

SHELLOPTS

A colon-separated list of enabled shell options. Each word in the list is a valid argument for the **-o** option to the **set** builtin command (see **SHELL BUILTIN COMMANDS** below). The options appearing in **SHELLOPTS** are those reported as *on* by **set -o**. If this variable is in the environment when **bash** starts up, each shell option in the list will be enabled before reading any startup files. This variable is read-only.

SHLVL

Incremented by one each time an instance of **bash** is started.

SRANDOM

This variable expands to a 32-bit pseudo-random number each time it is referenced. The random number generator is not linear on systems that support `/dev/urandom` or `arc4random`, so each returned number has no relationship to the numbers preceding it. The random number generator cannot be seeded, so assignments to this variable have no effect. If **SRANDOM** is unset, it loses its special properties, even if it is subsequently reset.

UID Expands to the user ID of the current user, initialized at shell startup. This variable is readonly.

The following variables are used by the shell. In some cases, **bash** assigns a default value to a variable; these cases are noted below.

BASH_COMPAT

The value is used to set the shell's compatibility level. See **SHELL COMPATIBILITY MODE** below for a description of the various compatibility levels and their effects. The value may be a decimal number (e.g., 4.2) or an integer (e.g., 42) corresponding to the desired compatibility level. If **BASH_COMPAT** is unset or set to the empty string, the compatibility level is set to the default for the current version. If **ASH_COMPAT** is set to a value that is not one of the valid compatibility levels, the shell prints an error message and sets the compatibility level to the default for the current version. The valid values correspond to the compatibility levels described below under **BASHCOMPATIBILITYMODE**. For example, 4.2 and 42 are valid values that correspond to the **compat42** **shop** option and set the compatibility level to 42. The current version is also a valid value.

BASH_ENV

If this parameter is set when **bash** is executing a shell script, its value is interpreted as a filename containing commands to initialize the shell, as in `%bashrc`. The value of **BASH_ENV** is subjected to parameter expansion, command substitution, and arithmetic expansion before being interpreted as a filename. **PATH** is not used to search for the resultant filename.

BASH_XTRACEFD

If set to an integer corresponding to a valid file descriptor, **bash** will write the trace output generated when **set -x** is enabled to that file descriptor. The file descriptor is closed when **BASH_XTRACEFD** is unset or assigned a new value. Unsetting **BASH_XTRACEFD** or assigning it the empty string causes the trace output to be sent to the standard error. Note that setting **BASH_XTRACEFD** to 2 (the standard error file descriptor) and then unsetting it will result in the standard error being closed.

CDPATH

The search path for the **cd** command. This is a colon-separated list of directories in which the shell looks for destination directories specified by the **cd** command. A sample value is `".:~:/usr"`.

CHILD_MAX

Set the number of exited child status values for the shell to remember. Bash will not allow this value to be decreased below a POSIX-mandated minimum, and there is a maximum value (currently 8192) that this may not exceed. The minimum value is system-dependent.

COLUMNS

Used by the **select** compound command to determine the terminal width when printing selection lists. Automatically set if the **checkwinsize** option is enabled or in an interactive shell upon receipt of a **SIGWINCH**.

COMPREPLY

An array variable from which **bash** reads the possible completions generated by a shell function invoked by the programmable completion facility (see **Programmable Completion** below). Each array element contains one possible completion.

EMACS

If **bash** finds this variable in the environment when the shell starts with value `t`, it assumes that the shell is running in an Emacs shell buffer and disables line editing.

ENV Expanded and executed similarly to **BASH_ENV** (see **INVOCATION** above) when an interactive shell is invoked in *posix mode*.

EXECIGNORE

A colon-separated list of shell patterns (see **Pattern Matching**) defining the list of filenames to be ignored by command search using **PATH**. Files whose full pathnames match one of these patterns are not considered executable files for the purposes of completion and command execution via **PATH** lookup. This does not affect the behavior of the **[**, **test**, and **[[** commands. Full pathnames in the command hash table are not subject to **EXECIGNORE**. Use this variable to ignore shared library files that have the executable bit set, but are not executable files. The pattern matching honors the setting of the **extglob** shell option.

FCEDIT

The default editor for the **fc** builtin command.

FIGNORE

A colon-separated list of suffixes to ignore when performing filename completion (see **READLINE** below). A filename whose suffix matches one of the entries in **FIGNORE** is excluded from the list of matched filenames. A sample value is "`.o:~`". (Quoting is needed when assigning a value to this variable, which contains tildes).

FUNCNEST

If set to a numeric value greater than 0, defines a maximum function nesting level. Function invocations that exceed this nesting level will cause the current command to abort.

GLOBIGNORE

A colon-separated list of patterns defining the set of file names to be ignored by pathname expansion. If a file name matched by a pathname expansion pattern also matches one of the patterns in **GLOBIGNORE**, it is removed from the list of matches.

HISTCONTROL

A colon-separated list of values controlling how commands are saved on the history list. If the list of values includes *ignorespace*, lines which begin with a **space** character are not saved in the history list. A value of *ignoredups* causes lines matching the previous history entry to not be saved. A value of *ignoreboth* is shorthand for *ignorespace* and *ignoredups*. A value of *erasedups* causes all previous lines matching the current line to be removed from the history list before that line is saved. Any value not in the above list is ignored. If **HISTCONTROL** is unset, or does not include a valid value, all lines read by the shell parser are saved on the history list, subject to the value of **HISTIGNORE**. The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of **HISTCONTROL**.

HISTFILE

The name of the file in which command history is saved (see **HISTORY** below). The default value is `~/.bash_history`. If unset, the command history is not saved when a shell exits.

HISTFILESIZE

The maximum number of lines contained in the history file. When this variable is assigned a value, the history file is truncated, if necessary, to contain no more than that number of lines by removing the oldest entries. The history file is also truncated to this size after writing it when a shell exits. If the value is 0, the history file is truncated to zero size. Non-numeric values and numeric values less than zero inhibit truncation. The shell sets the default value to the value of **HISTSIZE** after reading any startup files.

HISTIGNORE

A colon-separated list of patterns used to decide which command lines should be saved on the history list. Each pattern is anchored at the beginning of the line and must match the complete line (no implicit '*' is appended). Each pattern is tested against the line after the checks specified by **HISTCONTROL** are applied. In addition to the normal shell pattern matching characters, '&' matches the previous history line. '&' may be escaped using a backslash; the backslash is removed before attempting a match. The second and subsequent lines of a multi-line compound command are not tested, and are added to the history regardless of the value of **HISTIGNORE**. The pattern matching honors the setting of the **extglob** shell option.

HISTSIZE

The number of commands to remember in the command history (see **HISTORY** below). If the value is 0, commands are not saved in the history list. Numeric values less than zero result in

every command being saved on the history list (there is no limit). The shell sets the default value to 500 after reading any startup files.

HISTTIMEFORMAT

If this variable is set and not null, its value is used as a format string for *strftime*(3) to print the time stamp associated with each history entry displayed by the **history** builtin. If this variable is set, time stamps are written to the history file so they may be preserved across shell sessions. This uses the history comment character to distinguish timestamps from other history lines.

HOME

The home directory of the current user; the default argument for the **cd** builtin command. The value of this variable is also used when performing tilde expansion.

HOSTFILE

Contains the name of a file in the same format as */etc/hosts* that should be read when the shell needs to complete a hostname. The list of possible hostname completions may be changed while the shell is running; the next time hostname completion is attempted after the value is changed, **bash** adds the contents of the new file to the existing list. If **HOSTFILE** is set, but has no value, or does not name a readable file, **bash** attempts to read */etc/hosts* to obtain the list of possible hostname completions. When **HOSTFILE** is unset, the hostname list is cleared.

IFS The *Internal Field Separator* that is used for word splitting after expansion and to split lines into words with the **read** builtin command. The default value is “<space><tab><newline>”.

IGNOREEOF

Controls the action of an interactive shell on receipt of an **EOF** character as the sole input. If set, the value is the number of consecutive **EOF** characters which must be typed as the first characters on an input line before **bash** exits. If the variable exists but does not have a numeric value, or has no value, the default value is 10. If it does not exist, **EOF** signifies the end of input to the shell.

INPUTRC

The filename for the **readline** startup file, overriding the default of *~/.inputrc* (see **READLINE** below).

INSIDE_EMACS

If this variable appears in the environment when the shell starts, **bash** assumes that it is running inside an Emacs shell buffer and may disable line editing, depending on the value of **TERM**.

LANG Used to determine the locale category for any category not specifically selected with a variable starting with **LC_**.

LC_ALL

This variable overrides the value of **LANG** and any other **LC_** variable specifying a locale category.

LC_COLLATE

This variable determines the collation order used when sorting the results of pathname expansion, and determines the behavior of range expressions, equivalence classes, and collating sequences within pathname expansion and pattern matching.

LC_CTYPE

This variable determines the interpretation of characters and the behavior of character classes within pathname expansion and pattern matching.

LC_MESSAGES

This variable determines the locale used to translate double-quoted strings preceded by a \$.

LC_NUMERIC

This variable determines the locale category used for number formatting.

LC_TIME

This variable determines the locale category used for data and time formatting.

LINES Used by the **select** compound command to determine the column length for printing selection lists. Automatically set if the **checkwinsize** option is enabled or in an interactive shell upon receipt of a **SIGWINCH**.

MAIL If this parameter is set to a file or directory name and the **MAILPATH** variable is not set, **bash** informs the user of the arrival of mail in the specified file or Maildir-format directory.

MAILCHECK

Specifies how often (in seconds) **bash** checks for mail. The default is 60 seconds. When it is time to check for mail, the shell does so before displaying the primary prompt. If this variable is unset, or set to a value that is not a number greater than or equal to zero, the shell disables mail checking.

MAILPATH

A colon-separated list of filenames to be checked for mail. The message to be printed when mail arrives in a particular file may be specified by separating the filename from the message with a ‘?’.
When used in the text of the message, `$_` expands to the name of the current mailfile. Example:
`MAILPATH=/var/mail/bfox?"You have mail":~/shell-mail?"$_ has mail!"`
Bash can be configured to supply a default value for this variable (there is no value by default), but the location of the user mail files that it uses is system dependent (e.g., `/var/mail/$USER`).

OPTERR

If set to the value 1, **bash** displays error messages generated by the **getopts** builtin command (see **SHELL BUILTIN COMMANDS** below). **OPTERR** is initialized to 1 each time the shell is invoked or a shell script is executed.

PATH The search path for commands. It is a colon-separated list of directories in which the shell looks for commands (see **COMMAND EXECUTION** below). A zero-length (null) directory name in the value of **PATH** indicates the current directory. A null directory name may appear as two adjacent colons, or as an initial or trailing colon. The default path is system-dependent, and is set by the administrator who installs **bash**. A common value is `/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin`.

POSIXLY_CORRECT

If this variable is in the environment when **bash** starts, the shell enters *posix mode* before reading the startup files, as if the `--posix` invocation option had been supplied. If it is set while the shell is running, **bash** enables *posix mode*, as if the command `set -o posix` had been executed. When the shell enters *posix mode*, it sets this variable if it was not already set.

PROMPT_COMMAND

If this variable is set, and is an array, the value of each set element is executed as a command prior to issuing each primary prompt. If this is set but not an array variable, its value is used as a command to execute instead.

PROMPT_DIRTRIM

If set to a number greater than zero, the value is used as the number of trailing directory components to retain when expanding the `\w` and `\W` prompt string escapes (see **PROMPTING** below). Characters removed are replaced with an ellipsis.

PS0 The value of this parameter is expanded (see **PROMPTING** below) and displayed by interactive shells after reading a command and before the command is executed.

PS1 The value of this parameter is expanded (see **PROMPTING** below) and used as the primary prompt string. The default value is `"\$-\$\"`.

PS2 The value of this parameter is expanded as with **PS1** and used as the secondary prompt string. The default is >.

PS3 The value of this parameter is used as the prompt for the **select** command (see **SHELL GRAMMAR** above).

PS4 The value of this parameter is expanded as with **PS1** and the value is printed before each command **bash** displays during an execution trace. The first character of the expanded value of **PS4** is replicated multiple times, as necessary, to indicate multiple levels of indirection. The default is `+`.

SHELL

This variable expands to the full pathname to the shell. If it is not set when the shell starts, **bash** assigns to it the full pathname of the current user's login shell.

TIMEFORMAT

The value of this parameter is used as a format string specifying how the timing information for pipelines prefixed with the **time** reserved word should be displayed. The `%` character introduces an escape sequence that is expanded to a time value or other information. The escape sequences and their meanings are as follows; the braces denote optional portions.

%%	A literal % .
%[p][I]R	The elapsed time in seconds.
%[p][I]U	The number of CPU seconds spent in user mode.
%[p][I]S	The number of CPU seconds spent in system mode.
%P	The CPU percentage, computed as (%U + %S) / %R .

The optional *p* is a digit specifying the *precision*, the number of fractional digits after a decimal point. A value of 0 causes no decimal point or fraction to be output. At most three places after the decimal point may be specified; values of *p* greater than 3 are changed to 3. If *p* is not specified, the value 3 is used.

The optional **I** specifies a longer format, including minutes, of the form *MMmSS.FFs*. The value of *p* determines whether or not the fraction is included.

If this variable is not set, **bash** acts as if it had the value **\$'\nreal\t%3lR\nuser\t%3lU\nsys\t%3lS'**. If the value is null, no timing information is displayed. A trailing newline is added when the format string is displayed.

TMOUT

If set to a value greater than zero, **TMOUT** is treated as the default timeout for the **read** builtin. The **select** command terminates if input does not arrive after **TMOUT** seconds when input is coming from a terminal. In an interactive shell, the value is interpreted as the number of seconds to wait for a line of input after issuing the primary prompt. **Bash** terminates after waiting for that number of seconds if a complete line of input does not arrive.

TMPDIR

If set, **bash** uses its value as the name of a directory in which **bash** creates temporary files for the shell's use.

auto_resume

This variable controls how the shell interacts with the user and job control. If this variable is set, single word simple commands without redirections are treated as candidates for resumption of an existing stopped job. There is no ambiguity allowed; if there is more than one job beginning with the string typed, the job most recently accessed is selected. The *name* of a stopped job, in this context, is the command line used to start it. If set to the value *exact*, the string supplied must match the name of a stopped job exactly; if set to *substring*, the string supplied needs to match a substring of the name of a stopped job. The *substring* value provides functionality analogous to the **%?** job identifier (see **JOB CONTROL** below). If set to any other value, the supplied string must be a prefix of a stopped job's name; this provides functionality analogous to the **%string** job identifier.

histchars

The two or three characters which control history expansion and tokenization (see **HISTORY EXPANSION** below). The first character is the *history expansion* character, the character which signals the start of a history expansion, normally ‘!’. The second character is the *quick substitution* character, which is used as shorthand for re-running the previous command entered, substituting one string for another in the command. The default is ‘^’. The optional third character is the character which indicates that the remainder of the line is a comment when found as the first character of a word, normally ‘#’. The history comment character causes history substitution to be skipped for the remaining words on the line. It does not necessarily cause the shell parser to treat the rest of the line as a comment.

Arrays

Bash provides one-dimensional indexed and associative array variables. Any variable may be used as an indexed array; the **declare** builtin will explicitly declare an array. There is no maximum limit on the size of an array, nor any requirement that members be indexed or assigned contiguously. Indexed arrays are referenced using integers (including arithmetic expressions) and are zero-based; associative arrays are referenced using arbitrary strings. Unless otherwise noted, indexed array indices must be non-negative integers.

An indexed array is created automatically if any variable is assigned to using the syntax *name[subscript]=value*. The *subscript* is treated as an arithmetic expression that must evaluate to a number. To

explicitly declare an indexed array, use **declare -a name** (see **SHELL BUILTIN COMMANDS** below). **declare -a name[subscript]** is also accepted; the *subscript* is ignored.

Associative arrays are created using **declare -A name**.

Attributes may be specified for an array variable using the **declare** and **readonly** builtins. Each attribute applies to all members of an array.

Arrays are assigned to using compound assignments of the form *name*=(*value1* ... *valuen*), where each *value* may be of the form [subscript]=*string*. Indexed array assignments do not require anything but *string*. Each *value* in the list is expanded using all the shell expansions described below under **EXPANSION**. When assigning to indexed arrays, if the optional brackets and subscript are supplied, that index is assigned to; otherwise the index of the element assigned is the last index assigned to by the statement plus one. Indexing starts at zero.

When assigning to an associative array, the words in a compound assignment may be either assignment statements, for which the subscript is required, or a list of words that is interpreted as a sequence of alternating keys and values: *name*=(*key1 value1 key2 value2 ...*). These are treated identically to *name*=[*key1*]=*value1* [*key2*]=*value2* ...). The first word in the list determines how the remaining words are interpreted; all assignments in a list must be of the same type. When using key/value pairs, the keys may not be missing or empty; a final missing value is treated like the empty string.

This syntax is also accepted by the **declare** builtin. Individual array elements may be assigned to using the *name*[subscript]=*value* syntax introduced above. When assigning to an indexed array, if *name* is subscripted by a negative number, that number is interpreted as relative to one greater than the maximum index of *name*, so negative indices count back from the end of the array, and an index of -1 references the last element.

Any element of an array may be referenced using \${*name*[subscript]}. The braces are required to avoid conflicts with pathname expansion. If *subscript* is @ or *, the word expands to all members of *name*. These subscripts differ only when the word appears within double quotes. If the word is double-quoted, \${*name*[*]} expands to a single word with the value of each array member separated by the first character of the **IFS** special variable, and \${*name*@[]} expands each element of *name* to a separate word. When there are no array members, \${*name*@[]}} expands to nothing. If the double-quoted expansion occurs within a word, the expansion of the first parameter is joined with the beginning part of the original word, and the expansion of the last parameter is joined with the last part of the original word. This is analogous to the expansion of the special parameters * and @ (see **Special Parameters** above). \${#*name*[subscript]} expands to the length of \${*name*[subscript]}. If *subscript* is * or @, the expansion is the number of elements in the array. If the *subscript* used to reference an element of an indexed array evaluates to a number less than zero, it is interpreted as relative to one greater than the maximum index of the array, so negative indices count back from the end of the array, and an index of -1 references the last element.

Referencing an array variable without a subscript is equivalent to referencing the array with a subscript of 0. Any reference to a variable using a valid subscript is legal, and **bash** will create an array if necessary.

An array variable is considered set if a subscript has been assigned a value. The null string is a valid value.

It is possible to obtain the keys (indices) of an array as well as the values. \${!*name*@[]}} and \${!*name*[*]} expand to the indices assigned in array variable *name*. The treatment when in double quotes is similar to the expansion of the special parameters @ and * within double quotes.

The **unset** builtin is used to destroy arrays. **unset** *name*[subscript] destroys the array element at index *subscript*, for both indexed and associative arrays. Negative subscripts to indexed arrays are interpreted as described above. Unsetting the last element of an array variable does not unset the variable. **unset** *name*, where *name* is an array, or **unset** *name*[subscript], where *subscript* is * or @, removes the entire array.

When using a variable name with a subscript as an argument to a command, such as with **unset**, without using the word expansion syntax described above, the argument is subject to pathname expansion. If pathname expansion is not desired, the argument should be quoted.

The **declare**, **local**, and **readonly** builtins each accept a -a option to specify an indexed array and a -A option to specify an associative array. If both options are supplied, -A takes precedence. The **read** builtin

accepts a **-a** option to assign a list of words read from the standard input to an array. The **set** and **declare e** builtins display array values in a way that allows them to be reused as assignments.

EXPANSION

Expansion is performed on the command line after it has been split into words. There are seven kinds of expansion performed: *brace expansion*, *tilde expansion*, *parameter and variable expansion*, *command substitution*, *arithmetic expansion*, *word splitting*, and *pathname expansion*.

The order of expansions is: brace expansion; tilde expansion, parameter and variable expansion, arithmetic expansion, and command substitution (done in a left-to-right fashion); word splitting; and pathname expansion.

On systems that can support it, there is an additional expansion available: *process substitution*. This is performed at the same time as tilde, parameter, variable, and arithmetic expansion and command substitution.

After these expansions are performed, quote characters present in the original word are removed unless they have been quoted themselves (*quote removal*).

Only brace expansion, word splitting, and pathname expansion can increase the number of words of the expansion; other expansions expand a single word to a single word. The only exceptions to this are the expansions of "\$@" and "\${name[@]}", and, in most cases, \$* and \${name[*]} as explained above (see **PARAMETERS**).

Brace Expansion

Brace expansion is a mechanism by which arbitrary strings may be generated. This mechanism is similar to *pathname expansion*, but the filenames generated need not exist. Patterns to be brace expanded take the form of an optional *preamble*, followed by either a series of comma-separated strings or a sequence expression between a pair of braces, followed by an optional *postscript*. The preamble is prefixed to each string contained within the braces, and the postscript is then appended to each resulting string, expanding left to right.

Brace expansions may be nested. The results of each expanded string are not sorted; left to right order is preserved. For example, a{d,c,b}e expands into ‘ade ace abe’.

A sequence expression takes the form {*x..y[..incr]*}, where *x* and *y* are either integers or single characters, and *incr*, an optional increment, is an integer. When integers are supplied, the expression expands to each number between *x* and *y*, inclusive. Supplied integers may be prefixed with 0 to force each term to have the same width. When either *x* or *y* begins with a zero, the shell attempts to force all generated terms to contain the same number of digits, zero-padding where necessary. When characters are supplied, the expression expands to each character lexicographically between *x* and *y*, inclusive, using the default C locale. Note that both *x* and *y* must be of the same type. When the increment is supplied, it is used as the difference between each term. The default increment is 1 or -1 as appropriate.

Brace expansion is performed before any other expansions, and any characters special to other expansions are preserved in the result. It is strictly textual. **Bash** does not apply any syntactic interpretation to the context of the expansion or the text between the braces.

A correctly-formed brace expansion must contain unquoted opening and closing braces, and at least one unquoted comma or a valid sequence expression. Any incorrectly formed brace expansion is left unchanged. A { or , may be quoted with a backslash to prevent its being considered part of a brace expression. To avoid conflicts with parameter expansion, the string \${ is not considered eligible for brace expansion, and inhibits brace expansion until the closing }.

This construct is typically used as shorthand when the common prefix of the strings to be generated is longer than in the above example:

```
mkdir /usr/local/src/bash/{old,new,dist,bugs}
or
chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}}
```

Brace expansion introduces a slight incompatibility with historical versions of **sh**. **sh** does not treat opening or closing braces specially when they appear as part of a word, and preserves them in the output. **Bash**

removes braces from words as a consequence of brace expansion. For example, a word entered to **sh** as *file{1,2}* appears identically in the output. The same word is output as *file1 file2* after expansion by **bash**. If strict compatibility with **sh** is desired, start **bash** with the **+B** option or disable brace expansion with the **+B** option to the **set** command (see **SHELL BUILTIN COMMANDS** below).

Tilde Expansion

If a word begins with an unquoted tilde character (~), all of the characters preceding the first unquoted slash (or all characters, if there is no unquoted slash) are considered a *tilde-prefix*. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are treated as a possible *login name*. If this login name is the null string, the tilde is replaced with the value of the shell parameter **HOME**. If **HOME** is unset, the home directory of the user executing the shell is substituted instead. Otherwise, the tilde-prefix is replaced with the home directory associated with the specified login name.

If the tilde-prefix is a ~+, the value of the shell variable **PWD** replaces the tilde-prefix. If the tilde-prefix is a ~-, the value of the shell variable **OLDPWD**, if it is set, is substituted. If the characters following the tilde in the tilde-prefix consist of a number *N*, optionally prefixed by a + or a -, the tilde-prefix is replaced with the corresponding element from the directory stack, as it would be displayed by the **dirs** builtin invoked with the tilde-prefix as an argument. If the characters following the tilde in the tilde-prefix consist of a number without a leading + or -, + is assumed.

If the login name is invalid, or the tilde expansion fails, the word is unchanged.

Each variable assignment is checked for unquoted tilde-prefixes immediately following a : or the first =. In these cases, tilde expansion is also performed. Consequently, one may use filenames with tildes in assignments to **PATH**, **MAILPATH**, and **CDPATH**, and the shell assigns the expanded value.

Bash also performs tilde expansion on words satisfying the conditions of variable assignments (as described above under **PARAMETERS**) when they appear as arguments to simple commands. Bash does not do this, except for the *declaration* commands listed above, when in *posix mode*.

Parameter Expansion

The \$' character introduces parameter expansion, command substitution, or arithmetic expansion. The parameter name or symbol to be expanded may be enclosed in braces, which are optional but serve to protect the variable to be expanded from characters immediately following it which could be interpreted as part of the name.

When braces are used, the matching ending brace is the first '}' not escaped by a backslash or within a quoted string, and not within an embedded arithmetic expansion, command substitution, or parameter expansion.

\${parameter}

The value of *parameter* is substituted. The braces are required when *parameter* is a positional parameter with more than one digit, or when *parameter* is followed by a character which is not to be interpreted as part of its name. The *parameter* is a shell parameter as described above **PARAMETERS** or an array reference (**Arrays**).

If the first character of *parameter* is an exclamation point (!), and *parameter* is not a *nameref*, it introduces a level of indirection. **Bash** uses the value formed by expanding the rest of *parameter* as the new *parameter*; this is then expanded and that value is used in the rest of the expansion, rather than the expansion of the original *parameter*. This is known as *indirect expansion*. The value is subject to tilde expansion, parameter expansion, command substitution, and arithmetic expansion. If *parameter* is a *nameref*, this expands to the name of the parameter referenced by *parameter* instead of performing the complete indirect expansion. The exceptions to this are the expansions of \${!prefix*} and \${!name[@]} described below. The exclamation point must immediately follow the left brace in order to introduce indirection.

In each of the cases below, *word* is subject to tilde expansion, parameter expansion, command substitution, and arithmetic expansion.

When not performing substring expansion, using the forms documented below (e.g., :-), **bash** tests for a parameter that is unset or null. Omitting the colon results in a test only for a parameter that is unset.

`${parameter:-word}`

Use Default Values. If *parameter* is unset or null, the expansion of *word* is substituted. Otherwise, the value of *parameter* is substituted.

`${parameter:=word}`

Assign Default Values. If *parameter* is unset or null, the expansion of *word* is assigned to *parameter*. The value of *parameter* is then substituted. Positional parameters and special parameters may not be assigned to in this way.

`${parameter:?word}`

Display Error if Null or Unset. If *parameter* is null or unset, the expansion of *word* (or a message to that effect if *word* is not present) is written to the standard error and the shell, if it is not interactive, exits. Otherwise, the value of *parameter* is substituted.

`${parameter:+word}`

Use Alternate Value. If *parameter* is null or unset, nothing is substituted, otherwise the expansion of *word* is substituted.

`${parameter:offset}`

`${parameter:offset:length}`

Substring Expansion. Expands to up to *length* characters of the value of *parameter* starting at the character specified by *offset*. If *parameter* is @, an indexed array subscripted by @ or *, or an associative array name, the results differ as described below. If *length* is omitted, expands to the substring of the value of *parameter* starting at the character specified by *offset* and extending to the end of the value. *length* and *offset* are arithmetic expressions (see **ARITHMETIC EVALUATION** below).

If *offset* evaluates to a number less than zero, the value is used as an offset in characters from the end of the value of *parameter*. If *length* evaluates to a number less than zero, it is interpreted as an offset in characters from the end of the value of *parameter* rather than a number of characters, and the expansion is the characters between *offset* and that result. Note that a negative offset must be separated from the colon by at least one space to avoid being confused with the :- expansion.

If *parameter* is @, the result is *length* positional parameters beginning at *offset*. A negative *offset* is taken relative to one greater than the greatest positional parameter, so an offset of -1 evaluates to the last positional parameter. It is an expansion error if *length* evaluates to a number less than zero.

If *parameter* is an indexed array name subscripted by @ or *, the result is the *length* members of the array beginning with `${parameter[offset]}`. A negative *offset* is taken relative to one greater than the maximum index of the specified array. It is an expansion error if *length* evaluates to a number less than zero.

Substring expansion applied to an associative array produces undefined results.

Substring indexing is zero-based unless the positional parameters are used, in which case the indexing starts at 1 by default. If *offset* is 0, and the positional parameters are used, \$0 is prefixed to the list.

`${!prefix*}`

`${!prefix@}`

Names matching prefix. Expands to the names of variables whose names begin with *prefix*, separated by the first character of the IFS special variable. When @ is used and the expansion appears within double quotes, each variable name expands to a separate word.

`${!name[@]}`

`${!name[*]}`

List of array keys. If *name* is an array variable, expands to the list of array indices (keys) assigned in *name*. If *name* is not an array, expands to 0 if *name* is set and null otherwise. When @ is used and the expansion appears within double quotes, each key expands to a separate word.

`${#parameter}`

Parameter length. The length in characters of the value of *parameter* is substituted. If *parameter* is * or @, the value substituted is the number of positional parameters. If *parameter* is an array name subscripted by * or @, the value substituted is the number of elements in the array. If *parameter* is an indexed array name subscripted by a negative number, that number is interpreted as relative to one greater than the maximum index of *parameter*, so negative indices count back from the end of the array, and an index of -1 references the last element.

`${parameter##word}`
 `${parameter###word}`

Remove matching prefix pattern. The word *d* is expanded to produce a pattern just as in pathname expansion, and matched against the expanded value of *parameter* using the rules described under **Pattern Matching** below. If the pattern matches the beginning of the value of *parameter*, then the result of the expansion is the expanded value of *parameter* with the shortest matching pattern (the “#” case) or the longest matching pattern (the “##” case) deleted. If *parameter* is @ or *, the pattern removal operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with @ or *, the pattern removal operation is applied to each member of the array in turn, and the expansion is the resultant list.

`${parameter%word}`
 `${parameter%%word}`

Remove matching suffix pattern. The word *d* is expanded to produce a pattern just as in pathname expansion, and matched against the expanded value of *parameter* using the rules described under **Pattern Matching** below. If the pattern matches a trailing portion of the expanded value of *parameter*, then the result of the expansion is the expanded value of *parameter* with the shortest matching pattern (the “%” case) or the longest matching pattern (the “%%” case) deleted. If *parameter* is @ or *, the pattern removal operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with @ or *, the pattern removal operation is applied to each member of the array in turn, and the expansion is the resultant list.

`${parameter/pattern/string}`

Pattern substitution. The pattern is expanded to produce a pattern just as in pathname expansion, *Parameter* is expanded and the longest match of *pattern* against its value is replaced with *string*. The match is performed using the rules described under **Pattern Matching** below. If *pattern* begins with /, all matches of *pattern* are replaced with *string*. Normally only the first match is replaced. If *pattern* begins with #, it must match at the beginning of the expanded value of *parameter*. If *pattern* begins with %, it must match at the end of the expanded value of *parameter*. If *string* is null, matches of *pattern* are deleted and the / following *pattern* may be omitted. If the **nocasematch** shell option is enabled, the match is performed without regard to the case of alphabetic characters. If *parameter* is @ or *, the substitution operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with @ or *, the substitution operation is applied to each member of the array in turn, and the expansion is the resultant list.

`${parameter^pattern}`
 `${parameter^^pattern}`
 `${parameter,,pattern}`
 `${parameter,,,pattern}`

Case modification. This expansion modifies the case of alphabetic characters in *parameter*. The *pattern* is expanded to produce a pattern just as in pathname expansion. Each character in the expanded value of *parameter* is tested against *pattern*, and, if it matches the pattern, its case is converted. The pattern should not attempt to match more than one character. The ^ operator converts lowercase letters matching *pattern* to uppercase; the , operator converts matching uppercase letters to lowercase. The ^^ and ,, expansions convert each matched character in the expanded value; the ^ and , expansions match and convert only the first character in the expanded value. If *pattern* is

omitted, it is treated like a ?, which matches every character. If *parameter* is @ or *, the case modification operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with @ or *, the case modification operation is applied to each member of the array in turn, and the expansion is the resultant list.

`$(parameter@operator)`

Parameter transformation. The expansion is either a transformation of the value of *parameter* or information about *parameter* itself, depending on the value of *operator*. Each *operator* is a single letter:

- U** The expansion is a string that is the value of *parameter* with lowercase alphabetic characters converted to uppercase.
- u** The expansion is a string that is the value of *parameter* with the first character converted to uppercase, if it is alphabetic.
- L** The expansion is a string that is the value of *parameter* with uppercase alphabetic characters converted to lowercase.
- Q** The expansion is a string that is the value of *parameter* quoted in a format that can be reused as input.
- E** The expansion is a string that is the value of *parameter* with backslash escape sequences expanded as with the \$'...' quoting mechanism.
- P** The expansion is a string that is the result of expanding the value of *parameter* as if it were a prompt string (see **PROMPTING** below).
- A** The expansion is a string in the form of an assignment statement or **declare** command that, if evaluated, will recreate *parameter* with its attributes and value.
- K** Produces a possibly-quoted version of the value of *parameter*, except that it prints the values of indexed and associative arrays as a sequence of quoted key-value pairs (see **Arrays** above).
- a** The expansion is a string consisting of flag values representing *parameter*'s attributes.

If *parameter* is @ or *, the operation is applied to each positional parameter in turn, and the expansion is the resultant list. If *parameter* is an array variable subscripted with @ or *, the operation is applied to each member of the array in turn, and the expansion is the resultant list.

The result of the expansion is subject to word splitting and pathname expansion as described below.

Command Substitution

Command substitution allows the output of a command to replace the command name. There are two forms:

`$(command)`

or

``command``

Bash performs the expansion by executing *command* in a subshell environment and replacing the command substitution with the standard output of the command, with any trailing newlines deleted. Embedded newlines are not deleted, but they may be removed during word splitting. The command substitution `$(cat file)` can be replaced by the equivalent but faster `$(<file)`.

When the old-style backquote form of substitution is used, backslash retains its literal meaning except when followed by \$, ` , or \. The first backquote not preceded by a backslash terminates the command substitution. When using the `$(command)` form, all characters between the parentheses make up the command; none are treated specially.

Command substitutions may be nested. To nest when using the backquoted form, escape the inner backquotes with backslashes.

If the substitution appears within double quotes, word splitting and pathname expansion are not performed on the results.

Arithmetic Expansion

Arithmetic expansion allows the evaluation of an arithmetic expression and the substitution of the result. The format for arithmetic expansion is:

`$((expression))`

The old format `$[expression]` is deprecated and will be removed in upcoming versions of bash.

The *expression* is treated as if it were within double quotes, but a double quote inside the parentheses is not treated specially. All tokens in the expression undergo parameter and variable expansion, command substitution, and quote removal. The result is treated as the arithmetic expression to be evaluated. Arithmetic expansions may be nested.

The evaluation is performed according to the rules listed below under **ARITHMETIC EVALUATION**. If *expression* is invalid, **bash** prints a message indicating failure and no substitution occurs.

Process Substitution

Process substitution allows a process's input or output to be referred to using a filename. It takes the form of `<(list)` or `>(list)`. The *processlist* is run asynchronously, and its input or output appears as a filename. This filename is passed as an argument to the current command as the result of the expansion. If the `>(list)` form is used, writing to the file will provide input for *list*. If the `<(list)` form is used, the file passed as an argument should be read to obtain the output of *list*. Process substitution is supported on systems that support named pipes (*FIFOs*) or the `/dev/fd` method of naming open files.

When available, process substitution is performed simultaneously with parameter and variable expansion, command substitution, and arithmetic expansion.

Word Splitting

The shell scans the results of parameter expansion, command substitution, and arithmetic expansion that did not occur within double quotes for *word splitting*.

The shell treats each character of **IFS** as a delimiter, and splits the results of the other expansions into words using these characters as field terminators. If **IFS** is unset, or its value is exactly `<space><tab><newline>`, the default, then sequences of `<space>`, `<tab>`, and `<newline>` at the beginning and end of the results of the previous expansions are ignored, and any sequence of **IFS** characters not at the beginning or end serves to delimit words. If **IFS** has a value other than the default, then sequences of the whitespace characters **space**, **tab**, and **newline** are ignored at the beginning and end of the word, as long as the whitespace character is in the value of **IFS** (an **IFS** whitespace character). Any character in **IFS** that is not **IFS** whitespace, along with any adjacent **IFS** whitespace characters, delimits a field. A sequence of **IFS** whitespace characters is also treated as a delimiter. If the value of **IFS** is null, no word splitting occurs.

Explicit null arguments ("'" or '') are retained and passed to commands as empty strings. Unquoted implicit null arguments, resulting from the expansion of parameters that have no values, are removed. If a parameter with no value is expanded within double quotes, a null argument results and is retained and passed to a command as an empty string. When a quoted null argument appears as part of a word whose expansion is non-null, the null argument is removed. That is, the word `-d ''` becomes `-d` after word splitting and null argument removal.

Note that if no expansion occurs, no splitting is performed.

Pathname Expansion

After word splitting, unless the **-f** option has been set, **bash** scans each word for the characters *****, **?**, and **[**. If one of these characters appears, and is not quoted, then the word is regarded as a *pattern*, and replaced with an alphabetically sorted list of filenames matching the pattern (see **Pattern Matching** below). If no matching filenames are found, and the shell option **nullglob** is not enabled, the word is left unchanged. If the **nullglob** option is set, and no matches are found, the word is removed. If the **failglob** shell option is set, and no matches are found, an error message is printed and the command is not executed. If the shell option **nocaseglob** is enabled, the match is performed without regard to the case of alphabetic characters. Note that when using range expressions like `[a-z]` (see below), letters of the other case may be included, depending on the setting of **LC_COLLATE**. When a pattern is used for pathname expansion, the character `"."` at the start of a name or immediately following a slash must be matched explicitly, unless the shell

option **dotglob** is set. The filenames “.” and “..” must always be matched explicitly, even if **dotglob** is set. In other cases, the ‘‘.’’ character is not treated specially. When matching a pathname, the slash character must always be matched explicitly by a slash in the pattern, but in other matching contexts it can be matched by a special pattern character as described below under **Pattern Matching**. See the description of **shopt** below under **SHELL BUILTIN COMMANDS** for a description of the **nocaseglob**, **nullglob**, **failglob**, and **dotglob** shell options.

The **GLOBIGNORE** shell variable may be used to restrict the set of file names matching a *pattern*. If **GLOBIGNORE** is set, each matching file name that also matches one of the patterns in **GLOBIGNORE** is removed from the list of matches. If the **nocaseglob** option is set, the matching against the patterns in **GLOBIGNORE** is performed without regard to case. The filenames “.” and “..” are always ignored when **GLOBIGNORE** is set and not null. However, setting **GLOBIGNORE** to a non-null value has the effect of enabling the **dotglob** shell option, so all other filenames beginning with a “.” will match. To get the old behavior of ignoring filenames beginning with a “.”, make “.*” one of the patterns in **GLOBIGNORE**. The **dotglob** option is disabled when **GLOBIGNORE** is unset. The pattern matching honors the setting of the **extglob** shell option.

Pattern Matching

Any character that appears in a pattern, other than the special pattern characters described below, matches itself. The NUL character may not occur in a pattern. A backslash escapes the following character; the escaping backslash is discarded when matching. The special pattern characters must be quoted if they are to be matched literally.

The special pattern characters have the following meanings:

- * Matches any string, including the null string. When the **globstar** shell option is enabled, and * is used in a pathname expansion context, two adjacent *'s used as a single pattern will match all files and zero or more directories and subdirectories. If followed by a /, two adjacent *'s will match only directories and subdirectories.
- ? Matches any single character.
- [...] Matches any one of the enclosed characters. A pair of characters separated by a hyphen denotes a *range expression*; any character that falls between those two characters, inclusive, using the current locale's collating sequence and character set, is matched. If the first character following the [is a ! or a ^ then any character not enclosed is matched. The sorting order of characters in range expressions is determined by the current locale and the values of the **LC_COLLATE** or **LC_ALL** shell variables, if set. To obtain the traditional interpretation of range expressions, where [a-d] is equivalent to [abcd], set value of the **LC_ALL** shell variable to C, or enable the **globasciiranges** shell option. A – may be matched by including it as the first or last character in the set. A] may be matched by including it as the first character in the set.

Within [and], *character classes* can be specified using the syntax [:*class*:], where *class* is one of the following classes defined in the POSIX standard:

alnum alpha ascii blank cntrl digit graph lower print punct space upper word xdigit

A character class matches any character belonging to that class. The **word** character class matches letters, digits, and the character _.

Within [and], an *equivalence class* can be specified using the syntax [=c=], which matches all characters with the same collation weight (as defined by the current locale) as the character *c*.

Within [and], the syntax [.*symbol*.] matches the collating symbol *symbol*.

If the **extglob** shell option is enabled using the **shopt** builtin, several extended pattern matching operators are recognized. In the following description, a *pattern-list* is a list of one or more patterns separated by a |. Composite patterns may be formed using one or more of the following sub-patterns:

?(pattern-list)	Matches zero or one occurrence of the given patterns
*(pattern-list)	Matches zero or more occurrences of the given patterns
+(pattern-list)	Matches one or more occurrences of the given patterns
@(pattern-list)	Matches one of the given patterns
!(pattern-list)	Matches anything except one of the given patterns

Complicated extended pattern matching against long strings is slow, especially when the patterns contain alternations and the strings contain multiple matches. Using separate matches against shorter strings, or using arrays of strings instead of a single long string, may be faster.

Quote Removal

After the preceding expansions, all unquoted occurrences of the characters \, ', and " that did not result from one of the above expansions are removed.

REDIRECTION

Before a command is executed, its input and output may be *redirected* using a special notation interpreted by the shell. Redirection allows commands' file handles to be duplicated, opened, closed, made to refer to different files, and can change the files the command reads from and writes to. Redirection may also be used to modify file handles in the current shell execution environment. The following redirection operators may precede or appear anywhere within a *simple command* or may follow a *command*. Redirections are processed in the order they appear, from left to right.

Each redirection that may be preceded by a file descriptor number may instead be preceded by a word of the form {*varname*}. In this case, for each redirection operator except >&- and <&-, the shell will allocate a file descriptor greater than or equal to 10 and assign it to *varname*. If >&- or <&- is preceded by {*varname*}, the value of *varname* defines the file descriptor to close. If {*varname*} is supplied, the redirection persists beyond the scope of the command, allowing the shell programmer to manage the file descriptor himself.

In the following descriptions, if the file descriptor number is omitted, and the first character of the redirection operator is <, the redirection refers to the standard input (file descriptor 0). If the first character of the redirection operator is >, the redirection refers to the standard output (file descriptor 1).

The word following the redirection operator in the following descriptions, unless otherwise noted, is subjected to brace expansion, tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, quote removal, pathname expansion, and word splitting. If it expands to more than one word, **bash** reports an error.

Note that the order of redirections is significant. For example, the command

```
ls > dirlist 2>&1
```

directs both standard output and standard error to the file *dirlist*, while the command

```
ls 2>&1 > dirlist
```

directs only the standard output to file *dirlist*, because the standard error was duplicated from the standard output before the standard output was redirected to *dirlist*.

Bash handles several filenames specially when they are used in redirections, as described in the following table. If the operating system on which**bash** is running provides these special files, bash will use them; otherwise it will emulate them internally with the behavior described below.

/dev/fd/fd

If *fd* is a valid integer, file descriptor *fd* is duplicated.

/dev/stdin

File descriptor 0 is duplicated.

/dev/stdout

File descriptor 1 is duplicated.

/dev/stderr

File descriptor 2 is duplicated.

/dev/tcp/host/port

If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, **bash** attempts to open the corresponding TCP socket.

/dev/udp/host/port

If *host* is a valid hostname or Internet address, and *port* is an integer port number or service name, **bash** attempts to open the corresponding UDP socket.

A failure to open or create a file causes the redirection to fail.

Redirections using file descriptors greater than 9 should be used with care, as they may conflict with file descriptors the shell uses internally.

Note that the **exec** builtin command can make redirections take effect in the current shell.

Redirecting Input

Redirection of input causes the file whose name results from the expansion of *word* to be opened for reading on file descriptor *n*, or the standard input (file descriptor 0) if *n* is not specified.

The general format for redirecting input is:

[n]<word

Redirecting Output

Redirection of output causes the file whose name results from the expansion of *word* to be opened for writing on file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified. If the file does not exist it is created; if it does exist it is truncated to zero size.

The general format for redirecting output is:

[n]>word

If the redirection operator is **>**, and the **noclobber** option to the **set** builtin has been enabled, the redirection will fail if the file whose name results from the expansion of *word* exists and is a regular file. If the redirection operator is **>|**, or the redirection operator is **>** and the **noclobber** option to the **set** builtin command is not enabled, the redirection is attempted even if the file named by *word* exists.

Appending Redirected Output

Redirection of output in this fashion causes the file whose name results from the expansion of *word* to be opened for appending on file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified. If the file does not exist it is created.

The general format for appending output is:

[n]>>word

Redirecting Standard Output and Standard Error

This construct allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be redirected to the file whose name is the expansion of *word*.

There are two formats for redirecting standard output and standard error:

&>word

and

>&word

Of the two forms, the first is preferred. This is semantically equivalent to

>word 2>&1

When using the second form, *word* may not expand to a number or **-**. If it does, other redirection operators

apply (see **Duplicating File Descriptors** below) for compatibility reasons.

Appending Standard Output and Standard Error

This construct allows both the standard output (file descriptor 1) and the standard error output (file descriptor 2) to be appended to the file whose name is the expansion of *word*.

The format for appending standard output and standard error is:

&>>word

This is semantically equivalent to

>>word 2>&1

(see **Duplicating File Descriptors** below).

Here Documents

This type of redirection instructs the shell to read input from the current source until a line containing only *delimiter* (with no trailing blanks) is seen. All of the lines read up to that point are then used as the standard input (or file descriptor *n* if *n* is specified) for a command.

The format of here-documents is:

[*n*]<<[−]word
here-document
delimiter

No parameter and variable expansion, command substitution, arithmetic expansion, or pathname expansion is performed on *word*. If any part of *word* is quoted, the *delimiter* is the result of quote removal on *word*, and the lines in the here-document are not expanded. If *word* is unquoted, all lines of the here-document are subjected to parameter expansion, command substitution, and arithmetic expansion, the character sequence <newline> is ignored, and \ must be used to quote the characters \, \$, and `.

If the redirection operator is <<-, then all leading tab characters are stripped from input lines and the line containing *delimiter*. This allows here-documents within shell scripts to be indented in a natural fashion.

Here Strings

A variant of here documents, the format is:

[*n*]<<<*word*

The *word* undergoes tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, and quote removal. Pathname expansion and word splitting are not performed. The result is supplied as a single string, with a newline appended, to the command on its standard input (or file descriptor *n* if *n* is specified).

Duplicating File Descriptors

The redirection operator

[*n*]<&*word*

is used to duplicate input file descriptors. If *word* expands to one or more digits, the file descriptor denoted by *n* is made to be a copy of that file descriptor. If the digits in *word* do not specify a file descriptor open for input, a redirection error occurs. If *word* evaluates to −, file descriptor *n* is closed. If *n* is not specified, the standard input (file descriptor 0) is used.

The operator

[*n*]>&*word*

is used similarly to duplicate output file descriptors. If *n* is not specified, the standard output (file descriptor 1) is used. If the digits in *word* do not specify a file descriptor open for output, a redirection error occurs. If *word* evaluates to −, file descriptor *n* is closed. As a special case, if *n* is omitted, and *word* does not expand to one or more digits or −, the standard output and standard error are redirected as described previously.

Moving File Descriptors

The redirection operator

`[n]<&digit-`

moves the file descriptor *digit* to file descriptor *n*, or the standard input (file descriptor 0) if *n* is not specified. *digit* is closed after being duplicated to *n*.

Similarly, the redirection operator

`[n]>&digit-`

moves the file descriptor *digit* to file descriptor *n*, or the standard output (file descriptor 1) if *n* is not specified.

Opening File Descriptors for Reading and Writing

The redirection operator

`[n]<>word`

causes the file whose name is the expansion of *word* to be opened for both reading and writing on file descriptor *n*, or on file descriptor 0 if *n* is not specified. If the file does not exist, it is created.

ALIASES

Aliases allow a string to be substituted for a word when it is used as the first word of a simple command. The shell maintains a list of aliases that may be set and unset with the **alias** and **unalias** builtin commands (see **SHELL BUILTIN COMMANDS** below). The first word of each simple command, if unquoted, is checked to see if it has an alias. If so, that word is replaced by the text of the alias. The characters /, \$, `, and = and any of the shell *metacharacters* or quoting characters listed above may not appear in an alias name. The replacement text may contain any valid shell input, including shell metacharacters. The first word of the replacement text is tested for aliases, but a word that is identical to an alias being expanded is not expanded a second time. This means that one may alias **ls** to **ls -F**, for instance, and **bash** does not try to recursively expand the replacement text. If the last character of the alias value is a *blank*, then the next command word following the alias is also checked for alias expansion.

Aliases are created and listed with the **alias** command, and removed with the **unalias** command.

There is no mechanism for using arguments in the replacement text. If arguments are needed, a shell function should be used (see **FUNCTIONS** below).

Aliases are not expanded when the shell is not interactive, unless the **expand_aliases** shell option is set using **shopt** (see the description of **shopt** under **SHELL BUILTIN COMMANDS** below).

The rules concerning the definition and use of aliases are somewhat confusing. **Bash** always reads at least one complete line of input, and all lines that make up a compound command, before executing any of the commands on that line or the compound command. Aliases are expanded when a command is read, not when it is executed. Therefore, an alias definition appearing on the same line as another command does not take effect until the next line of input is read. The commands following the alias definition on that line are not affected by the new alias. This behavior is also an issue when functions are executed. Aliases are expanded when a function definition is read, not when the function is executed, because a function definition is itself a command. As a consequence, aliases defined in a function are not available until after that function is executed. To be safe, always put alias definitions on a separate line, and do not use **alias** in compound commands.

For almost every purpose, aliases are superseded by shell functions.

FUNCTIONS

A shell function, defined as described above under **SHELL GRAMMAR**, stores a series of commands for later execution. When the name of a shell function is used as a simple command name, the list of commands associated with that function name is executed. Functions are executed in the context of the current shell; no new process is created to interpret them (contrast this with the execution of a shell script). When a function is executed, the arguments to the function become the positional parameters during its execution. The special parameter # is updated to reflect the change. Special parameter 0 is unchanged. The first

element of the **FUNCNAME** variable is set to the name of the function while the function is executing.

All other aspects of the shell execution environment are identical between a function and its caller with these exceptions: the **DEBUG** and **RETURN** traps (see the description of the **trap** builtin under **SHELL BUILTIN COMMANDS** below) are not inherited unless the function has been given the **trace** attribute (see the description of the **declare** builtin below) or the **-o functrace** shell option has been enabled with the **set** builtin (in which case all functions inherit the **DEBUG** and **RETURN** traps), and the **ERR** trap is not inherited unless the **-o errtrace** shell option has been enabled.

Variables local to the function may be declared with the **local** builtin command. Ordinarily, variables and their values are shared between the function and its caller. If a variable is declared **local**, the variable's visible scope is restricted to that function and its children (including the functions it calls). Local variables "shadow" variables with the same name declared at previous scopes. For instance, a local variable declared in a function hides a global variable of the same name: references and assignments refer to the local variable, leaving the global variable unmodified. When the function returns, the global variable is once again visible.

The shell uses *dynamic scoping* to control a variable's visibility within functions. With dynamic scoping, visible variables and their values are a result of the sequence of function calls that caused execution to reach the current function. The value of a variable that a function sees depends on its value within its caller, if any, whether that caller is the "global" scope or another shell function. This is also the value that a local variable declaration "shadows", and the value that is restored when the function returns.

For example, if a variable *var* is declared as local in function *func1*, and *func1* calls another function *func2*, references to *var* made from within *func2* will resolve to the local variable *var* from *func1*, shadowing any global variable named *var*.

The **unset** builtin also acts using the same dynamic scope: if a variable is local to the current scope, **unset** will unset it; otherwise the **unset** will refer to the variable found in any calling scope as described above. If a variable at the current local scope is unset, it will remain so until it is reset in that scope or until the function returns. Once the function returns, any instance of the variable at a previous scope will become visible. If the **unset** acts on a variable at a previous scope, any instance of a variable with that name that had been shadowed will become visible.

The **FUNCNEST** variable, if set to a numeric value greater than 0, defines a maximum function nesting level. Function invocations that exceed the limit cause the entire command to abort.

If the builtin command **return** is executed in a function, the function completes and execution resumes with the next command after the function call. Any command associated with the **RETURN** trap is executed before execution resumes. When a function completes, the values of the positional parameters and the special parameter **#** are restored to the values they had prior to the function's execution.

Function names and definitions may be listed with the **-f** option to the **declare** or **typeset** builtin commands. The **-F** option to **declare** or **typeset** will list the function names only (and optionally the source file and line number, if the **extdebug** shell option is enabled). Functions may be exported so that subshells automatically have them defined with the **-f** option to the **export** builtin. A function definition may be deleted using the **-f** option to the **unset** builtin.

Functions may be recursive. The **FUNCNEST** variable may be used to limit the depth of the function call stack and restrict the number of function invocations. By default, no limit is imposed on the number of recursive calls.

ARITHMETIC EVALUATION

The shell allows arithmetic expressions to be evaluated, under certain circumstances (see the **let** and **declare** builtin commands, the **((** compound command, and **Arithmetic Expansion**). Evaluation is done in fixed-width integers with no check for overflow, though division by 0 is trapped and flagged as an error. The operators and their precedence, associativity, and values are the same as in the C language. The following list of operators is grouped into levels of equal-precedence operators. The levels are listed in order of decreasing precedence.

```

id++ id--           variable post-increment and post-decrement
- +      unary minus and plus
++id --id       variable pre-increment and pre-decrement
! ~      logical and bitwise negation
**      exponentiation
* / %   multiplication, division, remainder
+ -     addition, subtraction
<< >>  left and right bitwise shifts
<= >= < >    comparison
== !=   equality and inequality
&      bitwise AND
^      bitwise exclusive OR
|      bitwise OR
&&    logical AND
||      logical OR
expr?expr:expr  conditional operator
= *= /= %= += -= <<= >>= &= ^= |==
               assignment
expr1,expr2  comma

```

Shell variables are allowed as operands; parameter expansion is performed before the expression is evaluated. Within an expression, shell variables may also be referenced by name without using the parameter expansion syntax. A shell variable that is null or unset evaluates to 0 when referenced by name without using the parameter expansion syntax. The value of a variable is evaluated as an arithmetic expression when it is referenced, or when a variable which has been given the *integer* attribute using **declare -i** is assigned a value. A null value evaluates to 0. A shell variable need not have its *integer* attribute turned on to be used in an expression.

Integer constants follow the C language definition, without suffixes or character constants. Constants with a leading 0 are interpreted as octal numbers. A leading 0x or 0X denotes hexadecimal. Otherwise, numbers take the form [*base*#]*n*, where the optional *base* is a decimal number between 2 and 64 representing the arithmetic base, and *n* is a number in that base. If *base*# is omitted, then base 10 is used. When specifying *n*, if a non-digit is required, the digits greater than 9 are represented by the lowercase letters, the uppercase letters, @, and _, in that order. If *base* is less than or equal to 36, lowercase and uppercase letters may be used interchangeably to represent numbers between 10 and 35.

Operators are evaluated in order of precedence. Sub-expressions in parentheses are evaluated first and may override the precedence rules above.

CONDITIONAL EXPRESSIONS

Conditional expressions are used by the [[compound command and the **test** and [builtin commands to test file attributes and perform string and arithmetic comparisons. The **test** and [commands determine their behavior based on the number of arguments; see the descriptions of those commands for any other command-specific actions.

Expressions are formed from the following unary or binary primaries. **Bash** handles several filenames specially when they are used in expressions. If the operating system on which **bash** is running provides these special files, **bash** will use them; otherwise it will emulate them internally with this behavior: If any *file* argument to one of the primaries is of the form */dev/fd/n*, then file descriptor *n* is checked. If the *file* argument to one of the primaries is one of */dev/stdin*, */dev/stdout*, or */dev/stderr*, file descriptor 0, 1, or 2, respectively, is checked.

Unless otherwise specified, primaries that operate on files follow symbolic links and operate on the target

of the link, rather than the link itself.

When used with [[, the < and > operators sort lexicographically using the current locale. The **test** command sorts using ASCII ordering.

- a** *file* True if *file* exists.
- b** *file* True if *file* exists and is a block special file.
- c** *file* True if *file* exists and is a character special file.
- d** *file* True if *file* exists and is a directory.
- e** *file* True if *file* exists.
- f** *file* True if *file* exists and is a regular file.
- g** *file* True if *file* exists and is set-group-id.
- h** *file* True if *file* exists and is a symbolic link.
- k** *file* True if *file* exists and its “sticky” bit is set.
- p** *file* True if *file* exists and is a named pipe (FIFO).
- r** *file* True if *file* exists and is readable.
- s** *file* True if *file* exists and has a size greater than zero.
- t** *fd* True if file descriptor *fd* is open and refers to a terminal.
- u** *file* True if *file* exists and its set-user-id bit is set.
- w** *file* True if *file* exists and is writable.
- x** *file* True if *file* exists and is executable.
- G** *file* True if *file* exists and is owned by the effective group id.
- L** *file* True if *file* exists and is a symbolic link.
- N** *file* True if *file* exists and has been modified since it was last read.
- O** *file* True if *file* exists and is owned by the effective user id.
- S** *file* True if *file* exists and is a socket.
- file1* -ef *file2*** True if *file1* and *file2* refer to the same device and inode numbers.
- file1* -nt *file2*** True if *file1* is newer (according to modification date) than *file2*, or if *file1* exists and *file2* does not.
- file1* -ot *file2*** True if *file1* is older than *file2*, or if *file2* exists and *file1* does not.
- o** *optname* True if the shell option *optname* is enabled. See the list of options under the description of the **-o** option to the **set** builtin below.
- v** *varname* True if the shell variable *varname* is set (has been assigned a value).
- R** *varname* True if the shell variable *varname* is set and is a name reference.
- z** *string* True if the length of *string* is zero.
- string***
- n** *string* True if the length of *string* is non-zero.
- string1* == *string2***
- string1* = *string2*** True if the strings are equal. = should be used with the **test** command for POSIX conformance. When used with the [[command, this performs pattern matching as described above (**Compound Commands**)).
- string1* != *string2*** True if the strings are not equal.
- string1* < *string2*** True if *string1* sorts before *string2* lexicographically.

string1 > string2

True if *string1* sorts after *string2* lexicographically.

arg1 OP arg2

OP is one of **-eq**, **-ne**, **-lt**, **-le**, **-gt**, or **-ge**. These arithmetic binary operators return true if *arg1* is equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to *arg2*, respectively. *Arg1* and *arg2* may be positive or negative integers. When used with the [[command, *Arg1* and *Arg2* are evaluated as arithmetic expressions (see **ARITHMETIC EVALUATION** above).

SIMPLE COMMAND EXPANSION

When a simple command is executed, the shell performs the following expansions, assignments, and redirections, from left to right, in the following order.

1. The words that the parser has marked as variable assignments (those preceding the command name) and redirections are saved for later processing.
2. The words that are not variable assignments or redirections are expanded. If any words remain after expansion, the first word is taken to be the name of the command and the remaining words are the arguments.
3. Redirections are performed as described above under **REDIRECTION**.
4. The text after the = in each variable assignment undergoes tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal before being assigned to the variable.

If no command name results, the variable assignments affect the current shell environment. Otherwise, the variables are added to the environment of the executed command and do not affect the current shell environment. If any of the assignments attempts to assign a value to a readonly variable, an error occurs, and the command exits with a non-zero status.

If no command name results, redirections are performed, but do not affect the current shell environment. A redirection error causes the command to exit with a non-zero status.

If there is a command name left after expansion, execution proceeds as described below. Otherwise, the command exits. If one of the expansions contained a command substitution, the exit status of the command is the exit status of the last command substitution performed. If there were no command substitutions, the command exits with a status of zero.

COMMAND EXECUTION

After a command has been split into words, if it results in a simple command and an optional list of arguments, the following actions are taken.

If the command name contains no slashes, the shell attempts to locate it. If there exists a shell function by that name, that function is invoked as described above in **FUNCTIONS**. If the name does not match a function, the shell searches for it in the list of shell builtins. If a match is found, that builtin is invoked.

If the name is neither a shell function nor a builtin, and contains no slashes, **bash** searches each element of the **PATH** for a directory containing an executable file by that name. **Bash** uses a hash table to remember the full pathnames of executable files (see **hash** under **SHELL BUILTIN COMMANDS** below). A full search of the directories in **PATH** is performed only if the command is not found in the hash table. If the search is unsuccessful, the shell searches for a defined shell function named **command_not_found_handle**. If that function exists, it is invoked in a separate execution environment with the original command and the original command's arguments as its arguments, and the function's exit status becomes the exit status of that subshell. If that function is not defined, the shell prints an error message and returns an exit status of 127.

If the search is successful, or if the command name contains one or more slashes, the shell executes the named program in a separate execution environment. Argument 0 is set to the name given, and the remaining arguments to the command are set to the arguments given, if any.

If this execution fails because the file is not in executable format, and the file is not a directory, it is

assumed to be a *shell script*, a file containing shell commands. A subshell is spawned to execute it. This subshell reinitializes itself, so that the effect is as if a new shell had been invoked to handle the script, with the exception that the locations of commands remembered by the parent (see **hash** below under **SHELL BUILTIN COMMANDS**) are retained by the child.

If the program is a file beginning with **#!**, the remainder of the first line specifies an interpreter for the program. The shell executes the specified interpreter on operating systems that do not handle this executable format themselves. The arguments to the interpreter consist of a single optional argument following the interpreter name on the first line of the program, followed by the name of the program, followed by the command arguments, if any.

COMMAND EXECUTION ENVIRONMENT

The shell has an *execution environment*, which consists of the following:

- open files inherited by the shell at invocation, as modified by redirections supplied to the **exec** builtin
- the current working directory as set by **cd**, **pushd**, or **popd**, or inherited by the shell at invocation
- the file creation mode mask as set by **umask** or inherited from the shell's parent
- current traps set by **trap**
- shell parameters that are set by variable assignment or with **set** or inherited from the shell's parent in the environment
- shell functions defined during execution or inherited from the shell's parent in the environment
- options enabled at invocation (either by default or with command-line arguments) or by **set**
- options enabled by **shopt**
- shell aliases defined with **alias**
- various process IDs, including those of background jobs, the value of **\$\$**, and the value of **PPID**

When a simple command other than a builtin or shell function is to be executed, it is invoked in a separate execution environment that consists of the following. Unless otherwise noted, the values are inherited from the shell.

- the shell's open files, plus any modifications and additions specified by redirections to the command
- the current working directory
- the file creation mode mask
- shell variables and functions marked for export, along with variables exported for the command, passed in the environment
- traps caught by the shell are reset to the values inherited from the shell's parent, and traps ignored by the shell are ignored

A command invoked in this separate environment cannot affect the shell's execution environment.

Command substitution, commands grouped with parentheses, and asynchronous commands are invoked in a subshell environment that is a duplicate of the shell environment, except that traps caught by the shell are reset to the values that the shell inherited from its parent at invocation. Builtin commands that are invoked as part of a pipeline are also executed in a subshell environment. Changes made to the subshell environment cannot affect the shell's execution environment.

Subshells spawned to execute command substitutions inherit the value of the **-e** option from the parent shell. When not in *posix mode*, **bash** clears the **-e** option in such subshells.

If a command is followed by a **&** and job control is not active, the default standard input for the command is the empty file **/dev/null**. Otherwise, the invoked command inherits the file descriptors of the calling shell as modified by redirections.

ENVIRONMENT

When a program is invoked it is given an array of strings called the *environment*. This is a list of *name=value* pairs, of the form *name=value*.

The shell provides several ways to manipulate the environment. On invocation, the shell scans its own environment and creates a parameter for each name found, automatically marking it for *export* to child processes. Executed commands inherit the environment. The **export** and **declare -x** commands allow parameters and functions to be added to and deleted from the environment. If the value of a parameter in the environment is modified, the new value becomes part of the environment, replacing the old. The environment inherited by any executed command consists of the shell's initial environment, whose values may be modified in the shell, less any pairs removed by the **unset** command, plus any additions via the **export** and **declare -x** commands.

The environment for any *simple command* or function may be augmented temporarily by prefixing it with parameter assignments, as described above in **PARAMETERS**. These assignment statements affect only the environment seen by that command.

If the **-k** option is set (see the **set** builtin command below), then *all* parameter assignments are placed in the environment for a command, not just those that precede the command name.

When **bash** invokes an external command, the variable **_** is set to the full filename of the command and passed to that command in its environment.

EXIT STATUS

The exit status of an executed command is the value returned by the *waitpid* system call or equivalent function. Exit statuses fall between 0 and 255, though, as explained below, the shell may use values above 125 specially. Exit statuses from shell builtins and compound commands are also limited to this range. Under certain circumstances, the shell will use special values to indicate specific failure modes.

For the shell's purposes, a command which exits with a zero exit status has succeeded. An exit status of zero indicates success. A non-zero exit status indicates failure. When a command terminates on a fatal signal *N*, **bash** uses the value of 128+*N* as the exit status.

If a command is not found, the child process created to execute it returns a status of 127. If a command is found but is not executable, the return status is 126.

If a command fails because of an error during expansion or redirection, the exit status is greater than zero.

Shell builtin commands return a status of 0 (*true*) if successful, and non-zero (*false*) if an error occurs while they execute. All builtins return an exit status of 2 to indicate incorrect usage, generally invalid options or missing arguments.

Bash itself returns the exit status of the last command executed, unless a syntax error occurs, in which case it exits with a non-zero value. See also the **exit** builtin command below.

SIGNALS

When **bash** is interactive, in the absence of any traps, it ignores **SIGTERM** (so that **kill 0** does not kill an interactive shell), and **SIGINT** is caught and handled (so that the **wait** builtin is interruptible). In all cases, **bash** ignores **SIGQUIT**. If job control is in effect, **bash** ignores **SIGTTIN**, **SIGTTOU**, and **SIGTSTP**.

Non-builtin commands run by **bash** have signal handlers set to the values inherited by the shell from its parent. When job control is not in effect, asynchronous commands ignore **SIGINT** and **SIGQUIT** in addition to these inherited handlers. Commands run as a result of command substitution ignore the keyboard-generated job control signals **SIGTTIN**, **SIGTTOU**, and **SIGTSTP**.

The shell exits by default upon receipt of a **SIGHUP**. Before exiting, an interactive shell resends the **SIGHUP** to all jobs, running or stopped. Stopped jobs are sent **SIGCONT** to ensure that they receive the **SIGHUP**. To prevent the shell from sending the signal to a particular job, it should be removed from the jobs table with the **disown** builtin (see **SHELL BUILTIN COMMANDS** below) or marked to not receive **SIGHUP** using **disown -h**.

If the **huponexit** shell option has been set with **shopt**, **bash** sends a **SIGHUP** to all jobs when an interactive login shell exits.

If **bash** is waiting for a command to complete and receives a signal for which a trap has been set, the trap will not be executed until the command completes. When **bash** is waiting for an asynchronous command via the **wait** builtin, the reception of a signal for which a trap has been set will cause the **wait** builtin to return immediately with an exit status greater than 128, immediately after which the trap is executed.

JOB CONTROL

Job control refers to the ability to selectively stop (*suspend*) the execution of processes and continue (*resume*) their execution at a later point. A user typically employs this facility via an interactive interface supplied jointly by the operating system kernel's terminal driver and **bash**.

The shell associates a *job* with each pipeline. It keeps a table of currently executing jobs, which may be listed with the **jobs** command. When **bash** starts a job asynchronously (in the *background*), it prints a line that looks like:

```
[1] 25647
```

indicating that this job is job number 1 and that the process ID of the last process in the pipeline associated with this job is 25647. All of the processes in a single pipeline are members of the same job. **Bash** uses the *job* abstraction as the basis for job control.

To facilitate the implementation of the user interface to job control, the operating system maintains the notion of a *current terminal process group ID*. Members of this process group (processes whose process group ID is equal to the current terminal process group ID) receive keyboard-generated signals such as **SIGINT**. These processes are said to be in the *foreground*. *Background* processes are those whose process group ID differs from the terminal's; such processes are immune to keyboard-generated signals. Only foreground processes are allowed to read from or, if the user so specifies with **stty tostop**, write to the terminal. Background processes which attempt to read from (write to when **stty tostop** is in effect) the terminal are sent a **SIGTTIN** (**SIGTTOU**) signal by the kernel's terminal driver, which, unless caught, suspends the process.

If the operating system on which **bash** is running supports job control, **bash** contains facilities to use it. Typing the *suspend* character (typically **^Z**, Control-Z) while a process is running causes that process to be stopped and returns control to **bash**. Typing the *delayed suspend* character (typically **^Y**, Control-Y) causes the process to be stopped when it attempts to read input from the terminal, and control to be returned to **bash**. The user may then manipulate the state of this job, using the **bg** command to continue it in the background, the **fg** command to continue it in the foreground, or the **kill** command to kill it. A **^Z** takes effect immediately, and has the additional side effect of causing pending output and typeahead to be discarded.

There are a number of ways to refer to a job in the shell. The character **%** introduces a job specification (*jobspec*). Job number *n* may be referred to as **%n**. A job may also be referred to using a prefix of the name used to start it, or using a substring that appears in its command line. For example, **%ce** refers to a stopped job whose command name begins with **ce**. If a prefix matches more than one job, **bash** reports an error. Using **%?ce**, on the other hand, refers to any job containing the string **ce** in its command line. If the substring matches more than one job, **bash** reports an error. The symbols **%%** and **%+** refer to the shell's notion of the *current job*, which is the last job stopped while it was in the foreground or started in the background. The *previous job* may be referenced using **%-**. If there is only a single job, **%+** and **%-** can both be used to refer to that job. In output pertaining to jobs (e.g., the output of the **jobs** command), the current job is always flagged with a **+**, and the previous job with a **-**. A single **%** (with no accompanying job specification) also refers to the current job.

Simply naming a job can be used to bring it into the foreground: **%1** is a synonym for "**fg %1**", bringing job 1 from the background into the foreground. Similarly, "**%1 &**" resumes job 1 in the background, equivalent to "**bg %1**".

The shell learns immediately whenever a job changes state. Normally, **bash** waits until it is about to print a prompt before reporting changes in a job's status so as to not interrupt any other output. If the **-b** option to the **set** builtin command is enabled, **bash** reports such changes immediately. Any trap on **SIGCHLD** is executed for each child that exits.

If an attempt to exit **bash** is made while jobs are stopped (or, if the **checkjobs** shell option has been enabled using the **shopt** builtin, running), the shell prints a warning message, and, if the **checkjobs** option is enabled, lists the jobs and their statuses. The **jobs** command may then be used to inspect their status. If a second attempt to exit is made without an intervening command, the shell does not print another warning, and any stopped jobs are terminated.

When the shell is waiting for a job or process using the **wait** builtin, and job control is enabled, **wait** will return when the job changes state. The **-f** option causes **wait** to wait until the job or process terminates before returning.

PROMPTING

When executing interactively, **bash** displays the primary prompt **PS1** when it is ready to read a command, and the secondary prompt **PS2** when it needs more input to complete a command. **Bash** displays **PS0** after it reads a command but before executing it. **Bash** displays **PS4** as described above before tracing each command when the **-x** option is enabled. **Bash** allows these prompt strings to be customized by inserting a number of backslash-escaped special characters that are decoded as follows:

\a	an ASCII bell character (07)
\d	the date in "Weekday Month Date" format (e.g., "Tue May 26")
\D{format}	the <i>format</i> is passed to <i>strftime(3)</i> and the result is inserted into the prompt string; an empty <i>format</i> results in a locale-specific time representation. The braces are required
\e	an ASCII escape character (033)
\h	the hostname up to the first ‘:’
\H	the hostname
\j	the number of jobs currently managed by the shell
\l	the basename of the shell’s terminal device name
\n	newline
\r	carriage return
\s	the name of the shell, the basename of \$0 (the portion following the final slash)
\t	the current time in 24-hour HH:MM:SS format
\T	the current time in 12-hour HH:MM:SS format
\@	the current time in 12-hour am/pm format
\A	the current time in 24-hour HH:MM format
\u	the username of the current user
\v	the version of bash (e.g., 2.00)
\V	the release of bash , version + patch level (e.g., 2.00.0)
\w	the current working directory, with \$HOME abbreviated with a tilde (uses the value of the PROMPT_DIRTRIM variable)
\W	the basename of the current working directory, with \$HOME abbreviated with a tilde
\!	the history number of this command
\#	the command number of this command
\\$	if the effective UID is 0, a #, otherwise a \$
\nnn	the character corresponding to the octal number <i>nnn</i>
\\\	a backslash
\[begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\]	end a sequence of non-printing characters

The command number and the history number are usually different: the history number of a command is its position in the history list, which may include commands restored from the history file (see **HISTORY** below), while the command number is the position in the sequence of commands executed during the current shell session. After the string is decoded, it is expanded via parameter expansion, command substitution, arithmetic expansion, and quote removal, subject to the value of the **promptvars** shell option (see the description of the **shopt** command under **SHELL BUILTIN COMMANDS** below). This can have unwanted side effects if escaped portions of the string appear within command substitution or contain characters special to word expansion.

READLINE

This is the library that handles reading input when using an interactive shell, unless the **--noediting** option is given at shell invocation. Line editing is also used when using the **-e** option to the **read** builtin. By default, the line editing commands are similar to those of Emacs. A vi-style line editing interface is also available. Line editing can be enabled at any time using the **-o emacs** or **-o vi** options to the **set** builtin (see **SHELL BUILTIN COMMANDS** below). To turn off line editing after the shell is running, use the **+o emacs** or **+o vi** options to the **set** builtin.

Readline Notation

In this section, the Emacs-style notation is used to denote keystrokes. Control keys are denoted by C-key, e.g., C-n means Control-N. Similarly, *meta* keys are denoted by M-key, so M-x means Meta-X. (On keyboards without a *meta* key, M-x means ESC x, i.e., press the Escape key then the x key. This makes ESC the *meta prefix*. The combination M-C-x means ESC-Control-x, or press the Escape key then hold the Control key while pressing the x key.)

Readline commands may be given numeric *arguments*, which normally act as a repeat count. Sometimes, however, it is the sign of the argument that is significant. Passing a negative argument to a command that acts in the forward direction (e.g., **kill-line**) causes that command to act in a backward direction. Commands whose behavior with arguments deviates from this are noted below.

When a command is described as *killing* text, the text deleted is saved for possible future retrieval (*yanking*). The killed text is saved in a *kill ring*. Consecutive kills cause the text to be accumulated into one unit, which can be yanked all at once. Commands which do not kill text separate the chunks of text on the kill ring.

Readline Initialization

Readline is customized by putting commands in an initialization file (the *inputrc* file). The name of this file is taken from the value of the **INPUTRC** variable. If that variable is unset, the default is *%inputrc*. If that file does not exist or cannot be read, the ultimate default is */etc/inputrc*. When a program which uses the readline library starts up, the initialization file is read, and the key bindings and variables are set. There are only a few basic constructs allowed in the readline initialization file. Blank lines are ignored. Lines beginning with a # are comments. Lines beginning with a \$ indicate conditional constructs. Other lines denote key bindings and variable settings.

The default key-bindings may be changed with an *inputrc* file. Other programs that use this library may add their own commands and bindings.

For example, placing

M-Control-u: universal-argument

or

C-Meta-u: universal-argument

into the *inputrc* would make M-C-u execute the readline command *universal-argument*.

The following symbolic character names are recognized: *RUBOUT*, *DEL*, *ESC*, *LFD*, *NEWLINE*, *RET*, *RETURN*, *SPC*, *SPACE*, and *TAB*.

In addition to command names, readline allows keys to be bound to a string that is inserted when the key is pressed (a *macro*).

Readline Key Bindings

The syntax for controlling key bindings in the *inputrc* file is simple. All that is required is the name of the command or the text of a macro and a key sequence to which it should be bound. The name may be specified in one of two ways: as a symbolic key name, possibly with *Meta-* or *Control-* prefixes, or as a key sequence.

When using the form **keyname:function-name** or **macro**, *keyname* is the name of a key spelled out in English. For example:

Control-u: universal-argument

Meta-Rubout: backward-kill-word

Control-o: "> output"

In the above example, *C-u* is bound to the function **universal-argument**, *M-DEL* is bound to the function **backward-kill-word**, and *C-o* is bound to run the macro expressed on the right hand side (that is, to insert the text > output into the line).

In the second form, "**keyseq**":*function-name* or *macro*, **keyseq** differs from **keyname** above in that strings denoting an entire key sequence may be specified by placing the sequence within double quotes. Some GNU Emacs style key escapes can be used, as in the following example, but the symbolic character names are not recognized.

```
"\C-u": universal-argument
"\C-x\C-r": re-read-init-file
"\e[11~": "Function Key 1"
```

In this example, *C-u* is again bound to the function **universal-argument**. *C-x C-r* is bound to the function **re-read-init-file**, and *ESC / 11 ~* is bound to insert the text Function Key 1.

The full set of GNU Emacs style escape sequences is

\C-	control prefix
\M-	meta prefix
\e	an escape character
\	backslash
\"	literal "
\'	literal '

In addition to the GNU Emacs style escape sequences, a second set of backslash escapes is available:

\a	alert (bell)
\b	backspace
\d	delete
\f	form feed
\n	newline
\r	carriage return
\t	horizontal tab
\v	vertical tab
\nnn	the eight-bit character whose value is the octal value <i>nnn</i> (one to three digits)
\xHH	the eight-bit character whose value is the hexadecimal value <i>HH</i> (one or two hex digits)

When entering the text of a macro, single or double quotes must be used to indicate a macro definition. Unquoted text is assumed to be a function name. In the macro body, the backslash escapes described above are expanded. Backslash will quote any other character in the macro text, including " and '.

Bash allows the current readline key bindings to be displayed or modified with the **bind** builtin command. The editing mode may be switched during interactive use by using the **-o** option to the **set** builtin command (see **SHELL BUILTIN COMMANDS** below).

Readline Variables

Readline has variables that can be used to further customize its behavior. A variable may be set in the *inputrc* file with a statement of the form

```
set variable-name value
```

or using the **bind** builtin command (see **SHELL BUILTIN COMMANDS** below).

Except where noted, readline variables can take the values **On** or **Off** (without regard to case). Unrecognized variable names are ignored. When a variable value is read, empty or null values, "on" (case-insensitive), and "1" are equivalent to **On**. All other values are equivalent to **Off**. The variables and their default values are:

bell-style (audible)

Controls what happens when readline wants to ring the terminal bell. If set to **none**, readline never rings the bell. If set to **visible**, readline uses a visible bell if one is available. If set to **audible**,

readline attempts to ring the terminal's bell.

bind-tty-special-chars (On)

If set to **On**, readline attempts to bind the control characters treated specially by the kernel's terminal driver to their readline equivalents.

blink-matching-paren (Off)

If set to **On**, readline attempts to briefly move the cursor to an opening parenthesis when a closing parenthesis is inserted.

colored-completion-prefix (Off)

If set to **On**, when listing completions, readline displays the common prefix of the set of possible completions using a different color. The color definitions are taken from the value of the **LS_COLORS** environment variable.

colored-stats (Off)

If set to **On**, readline displays possible completions using different colors to indicate their file type. The color definitions are taken from the value of the **LS_COLORS** environment variable.

comment-begin ("#")

The string that is inserted when the readline **insert-comment** command is executed. This command is bound to **M-#** in emacs mode and to **#** in vi command mode.

completion-display-width (-1)

The number of screen columns used to display possible matches when performing completion. The value is ignored if it is less than 0 or greater than the terminal screen width. A value of 0 will cause matches to be displayed one per line. The default value is -1.

completion-ignore-case (Off)

If set to **On**, readline performs filename matching and completion in a case-insensitive fashion.

completion-map-case (Off)

If set to **On**, and **completion-ignore-case** is enabled, readline treats hyphens (-) and underscores (_) as equivalent when performing case-insensitive filename matching and completion.

completion-prefix-display-length (0)

The length in characters of the common prefix of a list of possible completions that is displayed without modification. When set to a value greater than zero, common prefixes longer than this value are replaced with an ellipsis when displaying possible completions.

completion-query-items (100)

This determines when the user is queried about viewing the number of possible completions generated by the **possible-completions** command. It may be set to any integer value greater than or equal to zero. If the number of possible completions is greater than or equal to the value of this variable, readline will ask whether or not the user wishes to view them; otherwise they are simply listed on the terminal.

convert-meta (On)

If set to **On**, readline will convert characters with the eighth bit set to an ASCII key sequence by stripping the eighth bit and prefixing an escape character (in effect, using escape as the *meta prefix*). The default is *On*, but readline will set it to *Off* if the locale contains eight-bit characters.

disable-completion (Off)

If set to **On**, readline will inhibit word completion. Completion characters will be inserted into the line as if they had been mapped to **self-insert**.

echo-control-characters (On)

When set to **On**, on operating systems that indicate they support it, readline echoes a character corresponding to a signal generated from the keyboard.

editing-mode (emacs)

Controls whether readline begins with a set of key bindings similar to *Emacs* or *vi*. **editing-mode** can be set to either **emacs** or **vi**.

emacs-mode-string (@)

If the *show-mode-in-prompt* variable is enabled, this string is displayed immediately before the last line of the primary prompt when emacs editing mode is active. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the \1 and \2 escapes to begin and end sequences of non-printing characters, which

can be used to embed a terminal control sequence into the mode string.

enable-bracketed-paste (On)

When set to **On**, readline will configure the terminal in a way that will enable it to insert each paste into the editing buffer as a single string of characters, instead of treating each character as if it had been read from the keyboard. This can prevent pasted characters from being interpreted as editing commands.

enable-keypad (Off)

When set to **On**, readline will try to enable the application keypad when it is called. Some systems need this to enable the arrow keys.

enable-meta-key (On)

When set to **On**, readline will try to enable any meta modifier key the terminal claims to support when it is called. On many terminals, the meta key is used to send eight-bit characters.

expand-tilde (Off)

If set to **On**, tilde expansion is performed when readline attempts word completion.

history-preserve-point (Off)

If set to **On**, the history code attempts to place point at the same location on each history line retrieved with **previous-history** or **next-history**.

history-size (unset)

Set the maximum number of history entries saved in the history list. If set to zero, any existing history entries are deleted and no new entries are saved. If set to a value less than zero, the number of history entries is not limited. By default, the number of history entries is set to the value of the **HISTSIZE** shell variable. If an attempt is made to set **history-size** to a non-numeric value, the maximum number of history entries will be set to 500.

horizontal-scroll-mode (Off)

When set to **On**, makes readline use a single line for display, scrolling the input horizontally on a single screen line when it becomes longer than the screen width rather than wrapping to a new line. This setting is automatically enabled for terminals of height 1.

input-meta (Off)

If set to **On**, readline will enable eight-bit input (that is, it will not strip the eighth bit from the characters it reads), regardless of what the terminal claims it can support. The name **meta-flag** is a synonym for this variable. The default is *Off*, but readline will set it to *On* if the locale contains eight-bit characters.

isearch-terminators (“C-[C-J]”)

The string of characters that should terminate an incremental search without subsequently executing the character as a command. If this variable has not been given a value, the characters *ESC* and *C-J* will terminate an incremental search.

keymap (emacs)

Set the current readline keymap. The set of valid keymap names is *emacs*, *emacs-standard*, *emacs-meta*, *emacs-ctlx*, *vi*, *vi-command*, and *vi-insert*. *vi* is equivalent to *vi-command*; *emacs* is equivalent to *emacs-standard*. The default value is *emacs*; the value of **editing-mode** also affects the default keymap.

keyseq-timeout (500)

Specifies the duration readline will wait for a character when reading an ambiguous key sequence (one that can form a complete key sequence using the input read so far, or can take additional input to complete a longer key sequence). If no input is received within the timeout, readline will use the shorter but complete key sequence. The value is specified in milliseconds, so a value of 1000 means that readline will wait one second for additional input. If this variable is set to a value less than or equal to zero, or to a non-numeric value, readline will wait until another key is pressed to decide which key sequence to complete.

mark-directories (On)

If set to **On**, completed directory names have a slash appended.

mark-modified-lines (Off)

If set to **On**, history lines that have been modified are displayed with a preceding asterisk (*).

mark-symlinked-directories (Off)

If set to **On**, completed names which are symbolic links to directories have a slash appended (subject to the value of **mark-directories**).

match-hidden-files (On)

This variable, when set to **On**, causes readline to match files whose names begin with a ‘.’ (hidden files) when performing filename completion. If set to **Off**, the leading ‘.’ must be supplied by the user in the filename to be completed.

menu-complete-display-prefix (Off)

If set to **On**, menu completion displays the common prefix of the list of possible completions (which may be empty) before cycling through the list.

output-meta (Off)

If set to **On**, readline will display characters with the eighth bit set directly rather than as a meta-prefixed escape sequence. The default is *Off*, but readline will set it to *On* if the locale contains eight-bit characters.

page-completions (On)

If set to **On**, readline uses an internal *more*-like pager to display a screenful of possible completions at a time.

print-completions-horizontally (Off)

If set to **On**, readline will display completions with matches sorted horizontally in alphabetical order, rather than down the screen.

revert-all-at-newline (Off)

If set to **On**, readline will undo all changes to history lines before returning when **accept-line** is executed. By default, history lines may be modified and retain individual undo lists across calls to **readline**.

show-all-if-ambiguous (Off)

This alters the default behavior of the completion functions. If set to **On**, words which have more than one possible completion cause the matches to be listed immediately instead of ringing the bell.

show-all-if-unmodified (Off)

This alters the default behavior of the completion functions in a fashion similar to **show-all-if-ambiguous**. If set to **On**, words which have more than one possible completion without any possible partial completion (the possible completions don't share a common prefix) cause the matches to be listed immediately instead of ringing the bell.

show-mode-in-prompt (Off)

If set to **On**, add a string to the beginning of the prompt indicating the editing mode: emacs, vi command, or vi insertion. The mode strings are user-settable (e.g., *emacs-mode-string*).

skip-completed-text (Off)

If set to **On**, this alters the default completion behavior when inserting a single match into the line. It's only active when performing completion in the middle of a word. If enabled, readline does not insert characters from the completion that match characters after point in the word being completed, so portions of the word following the cursor are not duplicated.

vi-cmd-mode-string ((cmd))

If the *show-mode-in-prompt* variable is enabled, this string is displayed immediately before the last line of the primary prompt when vi editing mode is active and in command mode. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the `\1` and `\2` escapes to begin and end sequences of non-printing characters, which can be used to embed a terminal control sequence into the mode string.

vi-ins-mode-string ((ins))

If the *show-mode-in-prompt* variable is enabled, this string is displayed immediately before the last line of the primary prompt when vi editing mode is active and in insertion mode. The value is expanded like a key binding, so the standard set of meta- and control prefixes and backslash escape sequences is available. Use the `\1` and `\2` escapes to begin and end sequences of non-printing characters, which can be used to embed a terminal control sequence into the mode string.

visible-stats (Off)

If set to **On**, a character denoting a file's type as reported by *stat(2)* is appended to the filename when listing possible completions.

Readline Conditional Constructs

Readline implements a facility similar in spirit to the conditional compilation features of the C preprocessor which allows key bindings and variable settings to be performed as the result of tests. There are four parser directives used.

\$if The **\$if** construct allows bindings to be made based on the editing mode, the terminal being used, or the application using readline. The text of the test, after any comparison operator, extends to the end of the line; unless otherwise noted, no characters are required to isolate it.

mode The **mode=** form of the **\$if** directive is used to test whether readline is in emacs or vi mode. This may be used in conjunction with the **set k eymap** command, for instance, to set bindings in the *emacs-standard* and *emacs-ctlx* keymaps only if readline is starting out in emacs mode.

term The **term=** form may be used to include terminal-specific key bindings, perhaps to bind the key sequences output by the terminal's function keys. The word on the right side of the **=** is tested against both the full name of the terminal and the portion of the terminal name before the first **-**. This allows *sun* to match both *sun* and *sun-cmd*, for instance.

version

The **version** test may be used to perform comparisons against specific readline versions. The **version** expands to the current readline version. The set of comparison operators includes **=**, **==**, **!=**, **<=**, **>=**, **<**, and **>**. The version number supplied on the right side of the operator consists of a major version number, an optional decimal point, and an optional minor version (e.g., **7.1**). If the minor version is omitted, it is assumed to be **0**. The operator may be separated from the string **version** and from the version number argument by whitespace.

application

The **application** construct is used to include application-specific settings. Each program using the readline library sets the *application name*, and an initialization file can test for a particular value. This could be used to bind key sequences to functions useful for a specific program. For instance, the following command adds a key sequence that quotes the current or previous word in **bash**:

```
$if Bash
# Quote the current or previous word
"\C-xq": "\eb\"ef\""
$endif
```

variable

The **variable** construct provides simple equality tests for readline variables and values. The permitted comparison operators are **=**, **==**, and **!=**. The variable name must be separated from the comparison operator by whitespace; the operator may be separated from the value on the right hand side by whitespace. Both string and boolean variables may be tested. Boolean variables must be tested against the values *on* and *off*.

\$endif This command, as seen in the previous example, terminates an **\$if** command.

\$else Commands in this branch of the **\$if** directive are executed if the test fails.

\$include

This directive takes a single filename as an argument and reads commands and bindings from that file. For example, the following directive would read */etc/inputrc*:

```
$include /etc/inputrc
```

Searching

Readline provides commands for searching through the command history (see **HISTORY** below) for lines containing a specified string. There are two search modes: *incremental* and *non-incremental*.

Incremental searches begin before the user has finished typing the search string. As each character of the search string is typed, readline displays the next entry from the history matching the string typed so far. An incremental search requires only as many characters as needed to find the desired history entry. The characters present in the value of the **isearch-terminators** variable are used to terminate an incremental search. If that variable has not been assigned a value the Escape and Control-J characters will terminate an incremental search. Control-G will abort an incremental search and restore the original line. When the search is terminated, the history entry containing the search string becomes the current line.

To find other matching entries in the history list, type Control-S or Control-R as appropriate. This will search backward or forward in the history for the next entry matching the search string typed so far. Any other key sequence bound to a readline command will terminate the search and execute that command. For instance, a *newline* will terminate the search and accept the line, thereby executing the command from the history list.

Readline remembers the last incremental search string. If two Control-Rs are typed without any intervening characters defining a new search string, any remembered search string is used.

Non-incremental searches read the entire search string before starting to search for matching history lines. The search string may be typed by the user or be part of the contents of the current line.

Readline Command Names

The following is a list of the names of the commands and the default key sequences to which they are bound. Command names without an accompanying key sequence are unbound by default. In the following descriptions, *point* refers to the current cursor position, and *mark* refers to a cursor position saved by the **set-mark** command. The text between the point and mark is referred to as the *region*.

Commands for Moving

beginning-of-line (C-a)

Move to the start of the current line.

end-of-line (C-e)

Move to the end of the line.

forward-char (C-f)

Move forward a character.

backward-char (C-b)

Move back a character.

forward-word (M-f)

Move forward to the end of the next word. Words are composed of alphanumeric characters (letters and digits).

backward-word (M-b)

Move back to the start of the current or previous word. Words are composed of alphanumeric characters (letters and digits).

shell-forward-word

Move forward to the end of the next word. Words are delimited by non-quoted shell metacharacters.

shell-backward-word

Move back to the start of the current or previous word. Words are delimited by non-quoted shell metacharacters.

previous-screen-line

Attempt to move point to the same physical screen column on the previous physical screen line. This will not have the desired effect if the current Readline line does not take up more than one physical line or if point is not greater than the length of the prompt plus the screen width.

next-screen-line

Attempt to move point to the same physical screen column on the next physical screen line. This will not have the desired effect if the current Readline line does not take up more than one physical

line or if the length of the current Readline line is not greater than the length of the prompt plus the screen width.

clear–display (M–C–I)

Clear the screen and, if possible, the terminal's scrollback buffer, then redraw the current line, leaving the current line at the top of the screen.

clear–screen (C–I)

Clear the screen, then redraw the current line, leaving the current line at the top of the screen.

With an argument, refresh the current line without clearing the screen.

redraw–current–line

Refresh the current line.

Commands for Manipulating the History

accept–line (Newline, Return)

Accept the line regardless of where the cursor is. If this line is non-empty, add it to the history list according to the state of the **HISTCONTROL** variable. If the line is a modified history line, then restore the history line to its original state.

previous–history (C–p)

Fetch the previous command from the history list, moving back in the list.

next–history (C–n)

Fetch the next command from the history list, moving forward in the list.

beginning–of–history (M–<)

Move to the first line in the history.

end–of–history (M–>)

Move to the end of the input history, i.e., the line currently being entered.

reverse–search–history (C–r)

Search backward starting at the current line and moving 'up' through the history as necessary. This is an incremental search.

forward–search–history (C–s)

Search forward starting at the current line and moving 'down' through the history as necessary. This is an incremental search.

non–incremental–reverse–search–history (M–p)

Search backward through the history starting at the current line using a non-incremental search for a string supplied by the user.

non–incremental–forward–search–history (M–n)

Search forward through the history using a non-incremental search for a string supplied by the user.

history–search–forward

Search forward through the history for the string of characters between the start of the current line and the point. This is a non-incremental search.

history–search–backward

Search backward through the history for the string of characters between the start of the current line and the point. This is a non-incremental search.

history–substring–search–backward

Search backward through the history for the string of characters between the start of the current line and the current cursor position (the *point*). The search string may match anywhere in a history line. This is a non-incremental search.

history–substring–search–forward

Search forward through the history for the string of characters between the start of the current line and the point. The search string may match anywhere in a history line. This is a non-incremental search.

yank–nth–arg (M–C–y)

Insert the first argument to the previous command (usually the second word on the previous line) at point. With an argument *n*, insert the *n*th word from the previous command (the words in the previous command begin with word 0). A negative argument inserts the *n*th word from the end of the previous command. Once the argument *n* is computed, the argument is extracted as if the "!"

history expansion had been specified.

yank–last–arg (M–., M–_)

Insert the last argument to the previous command (the last word of the previous history entry).

With a numeric argument, behave exactly like **yank–nth–arg**. Successive calls to **yank–last–arg** move back through the history list, inserting the last word (or the word specified by the argument to the first call) of each line in turn. Any numeric argument supplied to these successive calls determines the direction to move through the history. A negative argument switches the direction through the history (back or forward). The history expansion facilities are used to extract the last word, as if the "\$!" history expansion had been specified.

shell–expand–line (M–C–e)

Expand the line as the shell does. This performs alias and history expansion as well as all of the shell word expansions. See **HISTORY EXPANSION** below for a description of history expansion.

history–expand–line (M–^)

Perform history expansion on the current line. See **HISTORY EXPANSION** below for a description of history expansion.

magic–space

Perform history expansion on the current line and insert a space. See **HISTORY EXPANSION** below for a description of history expansion.

alias–expand–line

Perform alias expansion on the current line. See **ALIASES** above for a description of alias expansion.

history–and–alias–expand–line

Perform history and alias expansion on the current line.

insert–last–argument (M–., M–_)

A synonym for **yank–last–arg**.

operate–and–get–next (C–o)

Accept the current line for execution and fetch the next line relative to the current line from the history for editing. A numeric argument, if supplied, specifies the history entry to use instead of the current line.

edit–and–execute–command (C–x C–e)

Invoke an editor on the current command line, and execute the result as shell commands. **Bash** attempts to invoke **\$VISUAL**, **\$EDITOR**, and *emacs* as the editor, in that order.

Commands for Changing Text

end–of–file (usually C–d)

The character indicating end-of-file as set, for example, by **stty**. If this character is read when there are no characters on the line, and point is at the beginning of the line, Readline interprets it as the end of input and returns **EOF**.

delete–char (C–d)

Delete the character at point. If this function is bound to the same character as the **tt** **EOF** character, as **C–d** commonly is, see above for the effects.

backward–delete–char (Rubout)

Delete the character behind the cursor. When given a numeric argument, save the deleted text on the kill ring.

forward–backward–delete–char

Delete the character under the cursor, unless the cursor is at the end of the line, in which case the character behind the cursor is deleted.

quoted–insert (C–q, C–v)

Add the next character typed to the line verbatim. This is how to insert characters like **C–q**, for example.

tab–insert (C–v TAB)

Insert a tab character.

self–insert (a, b, A, 1, !, ...)

Insert the character typed.

transpose-chars (C-t)

Drag the character before point forward over the character at point, moving point forward as well. If point is at the end of the line, then this transposes the two characters before point. Negative arguments have no effect.

transpose-words (M-t)

Drag the word before point past the word after point, moving point over that word as well. If point is at the end of the line, this transposes the last two words on the line.

upcase-word (M-u)

Uppercase the current (or following) word. With a negative argument, uppercase the previous word, but do not move point.

downcase-word (M-l)

Lowercase the current (or following) word. With a negative argument, lowercase the previous word, but do not move point.

capitalize-word (M-c)

Capitalize the current (or following) word. With a negative argument, capitalize the previous word, but do not move point.

overwrite-mode

Toggle overwrite mode. With an explicit positive numeric argument, switches to overwrite mode. With an explicit non-positive numeric argument, switches to insert mode. This command affects only **emacs** mode; **vi** mode does overwrite differently. Each call to `readline()` starts in insert mode. In overwrite mode, characters bound to **self-insert** replace the text at point rather than pushing the text to the right. Characters bound to **backward-delete-char** replace the character before point with a space. By default, this command is unbound.

Killing and Yanking**kill-line (C-k)**

Kill the text from point to the end of the line.

backward-kill-line (C-x Rubout)

Kill backward to the beginning of the line.

unix-line-discard (C-u)

Kill backward from point to the beginning of the line. The killed text is saved on the kill-ring.

kill-whole-line

Kill all characters on the current line, no matter where point is.

kill-word (M-d)

Kill from point to the end of the current word, or if between words, to the end of the next word.

Word boundaries are the same as those used by **forward-word**.

backward-kill-word (M-Rubout)

Kill the word behind point. Word boundaries are the same as those used by **backward-word**.

shell-kill-word

Kill from point to the end of the current word, or if between words, to the end of the next word.

Word boundaries are the same as those used by **shell-forward-word**.

shell-backward-kill-word

Kill the word behind point. Word boundaries are the same as those used by **shell-backward-word**.

unix-word-rubout (C-w)

Kill the word behind point, using white space as a word boundary. The killed text is saved on the kill-ring.

unix-filename-rubout

Kill the word behind point, using white space and the slash character as the word boundaries. The killed text is saved on the kill-ring.

delete-horizontal-space (M-|)

Delete all spaces and tabs around point.

kill-region

Kill the text in the current region.

copy-region-as-kill

Copy the text in the region to the kill buffer.

copy-backward-word

Copy the word before point to the kill buffer. The word boundaries are the same as **backward-word**.

copy-forward-word

Copy the word following point to the kill buffer. The word boundaries are the same as **forward-word**.

yank (C-y)

Yank the top of the kill ring into the buffer at point.

yank-pop (M-y)

Rotate the kill ring, and yank the new top. Only works following **yank** or **yank-pop**.

Numeric Arguments**digit-argument (M-0, M-1, ..., M--)**

Add this digit to the argument already accumulating, or start a new argument. M-- starts a negative argument.

universal-argument

This is another way to specify an argument. If this command is followed by one or more digits, optionally with a leading minus sign, those digits define the argument. If the command is followed by digits, executing **universal-argument** again ends the numeric argument, but is otherwise ignored. As a special case, if this command is immediately followed by a character that is neither a digit nor minus sign, the argument count for the next command is multiplied by four. The argument count is initially one, so executing this function the first time makes the argument count four, a second time makes the argument count sixteen, and so on.

Completing**complete (TAB)**

Attempt to perform completion on the text before point. **Bash** attempts completion treating the text as a variable (if the text begins with \$), username (if the text begins with ~), hostname (if the text begins with @), or command (including aliases and functions) in turn. If none of these produces a match, filename completion is attempted.

possible-completions (M-?)

List the possible completions of the text before point.

insert-completions (M-*)

Insert all completions of the text before point that would have been generated by **possible-completions**.

menu-complete

Similar to **complete**, but replaces the word to be completed with a single match from the list of possible completions. Repeated execution of **menu-complete** steps through the list of possible completions, inserting each match in turn. At the end of the list of completions, the bell is rung (subject to the setting of **bell-style**) and the original text is restored. An argument of *n* moves *n* positions forward in the list of matches; a negative argument may be used to move backward through the list. This command is intended to be bound to **TAB**, but is unbound by default.

menu-complete-backward

Identical to **menu-complete**, but moves backward through the list of possible completions, as if **menu-complete** had been given a negative argument. This command is unbound by default.

delete-char-or-list

Deletes the character under the cursor if not at the beginning or end of the line (like **delete-char**). If at the end of the line, behaves identically to **possible-completions**. This command is unbound by default.

complete-filename (M-/)

Attempt filename completion on the text before point.

possible-filename-completions (C-x /)

List the possible completions of the text before point, treating it as a filename.

complete-username (M-~)

Attempt completion on the text before point, treating it as a username.

possible-username-completions (C-x ~)

List the possible completions of the text before point, treating it as a username.

complete-variable (M-\$)

Attempt completion on the text before point, treating it as a shell variable.

possible-variable-completions (C-x \$)

List the possible completions of the text before point, treating it as a shell variable.

complete-hostname (M-@)

Attempt completion on the text before point, treating it as a hostname.

possible-hostname-completions (C-x @)

List the possible completions of the text before point, treating it as a hostname.

complete-command (M-!)

Attempt completion on the text before point, treating it as a command name. Command completion attempts to match the text against aliases, reserved words, shell functions, shell builtins, and finally executable filenames, in that order.

possible-command-completions (C-x !)

List the possible completions of the text before point, treating it as a command name.

dynamic-complete-history (M-TAB)

Attempt completion on the text before point, comparing the text against lines from the history list for possible completion matches.

dabbrev-expand

Attempt menu completion on the text before point, comparing the text against lines from the history list for possible completion matches.

complete-into-braces (M-{)

Perform filename completion and insert the list of possible completions enclosed within braces so the list is available to the shell (see **Brace Expansion** above).

Keyboard Macros**start-kbd-macro (C-x ()**

Begin saving the characters typed into the current keyboard macro.

end-kbd-macro (C-x))

Stop saving the characters typed into the current keyboard macro and store the definition.

call-last-kbd-macro (C-x e)

Re-execute the last keyboard macro defined, by making the characters in the macro appear as if typed at the keyboard.

print-last-kbd-macro ()

Print the last keyboard macro defined in a format suitable for the *inputrc* file.

Miscellaneous**re-read-init-file (C-x C-r)**

Read in the contents of the *inputrc* file, and incorporate any bindings or variable assignments found there.

abort (C-g)

Abort the current editing command and ring the terminal's bell (subject to the setting of **bell-style**).

do-lowercase-version (M-A, M-B, M-x, ...)

If the metabified character *x* is uppercase, run the command that is bound to the corresponding metabified lowercase character. The behavior is undefined if *x* is already lowercase.

prefix-meta (ESC)

Metabify the next character typed. **ESC f** is equivalent to **Meta-f**.

undo (C-, C-x C-u)

Incremental undo, separately remembered for each line.

revert-line (M-r)

Undo all changes made to this line. This is like executing the **undo** command enough times to return the line to its initial state.

tilde-expand (M-&)

Perform tilde expansion on the current word.

set-mark (C-@, M-<space>)

Set the mark to the point. If a numeric argument is supplied, the mark is set to that position.

exchange-point-and-mark (C-x C-x)

Swap the point with the mark. The current cursor position is set to the saved position, and the old cursor position is saved as the mark.

character-search (C-])

A character is read and point is moved to the next occurrence of that character. A negative count searches for previous occurrences.

character-search-backward (M-C-])

A character is read and point is moved to the previous occurrence of that character. A negative count searches for subsequent occurrences.

skip-csi-sequence

Read enough characters to consume a multi-key sequence such as those defined for keys like Home and End. Such sequences begin with a Control Sequence Indicator (CSI), usually ESC-[. If this sequence is bound to "\|", keys producing such sequences will have no effect unless explicitly bound to a readline command, instead of inserting stray characters into the editing buffer. This is unbound by default, but usually bound to ESC-[.

insert-comment (M-#)

Without a numeric argument, the value of the readline **comment-begin** variable is inserted at the beginning of the current line. If a numeric argument is supplied, this command acts as a toggle: if the characters at the beginning of the line do not match the value of **comment-begin**, the value is inserted, otherwise the characters in **comment-begin** are deleted from the beginning of the line. In either case, the line is accepted as if a newline had been typed. The default value of **comment-begin** causes this command to make the current line a shell comment. If a numeric argument causes the comment character to be removed, the line will be executed by the shell.

glob-complete-word (M-g)

The word before point is treated as a pattern for pathname expansion, with an asterisk implicitly appended. This pattern is used to generate a list of matching filenames for possible completions.

glob-expand-word (C-x *)

The word before point is treated as a pattern for pathname expansion, and the list of matching filenames is inserted, replacing the word. If a numeric argument is supplied, an asterisk is appended before pathname expansion.

glob-list-expansions (C-x g)

The list of expansions that would have been generated by **glob-expand-word** is displayed, and the line is redrawn. If a numeric argument is supplied, an asterisk is appended before pathname expansion.

dump-functions

Print all of the functions and their key bindings to the readline output stream. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

dump-variables

Print all of the settable readline variables and their values to the readline output stream. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

dump-macros

Print all of the readline key sequences bound to macros and the strings they output. If a numeric argument is supplied, the output is formatted in such a way that it can be made part of an *inputrc* file.

display-shell-version (C-x C-v)

Display version information about the current instance of **bash**.

Programmable Completion

When word completion is attempted for an argument to a command for which a completion specification (*compspec*) has been defined using the **complete** builtin (see **SHELL BUILTIN COMMANDS** below), the

programmable completion facilities are invoked.

First, the command name is identified. If the command word is the empty string (completion attempted at the beginning of an empty line), any compspec defined with the **-E** option to **complete** is used. If a compspec has been defined for that command, the compspec is used to generate the list of possible completions for the word. If the command word is a full pathname, a compspec for the full pathname is searched for first. If no compspec is found for the full pathname, an attempt is made to find a compspec for the portion following the final slash. If those searches do not result in a compspec, any compspec defined with the **-D** option to **complete** is used as the default. If there is no default compspec, **bash** attempts alias expansion on the command word as a final resort, and attempts to find a compspec for the command word from any successful expansion.

Once a compspec has been found, it is used to generate the list of matching words. If a compspec is not found, the default **bash** completion as described above under **Completing** is performed.

First, the actions specified by the compspec are used. Only matches which are prefixed by the word being completed are returned. When the **-f** or **-d** option is used for filename or directory name completion, the shell variable **IGNORE** is used to filter the matches.

Any completions specified by a pathname expansion pattern to the **-G** option are generated next. The words generated by the pattern need not match the word being completed. The **GLOBIGNORE** shell variable is not used to filter the matches, but the **IGNORE** variable is used.

Next, the string specified as the argument to the **-W** option is considered. The string is first split using the characters in the **IFS** special variable as delimiters. Shell quoting is honored. Each word is then expanded using brace expansion, tilde expansion, parameter and variable expansion, command substitution, and arithmetic expansion, as described above under **EXPANSION**. The results are split using the rules described above under **Word Splitting**. The results of the expansion are prefix-matched against the word being completed, and the matching words become the possible completions.

After these matches have been generated, any shell function or command specified with the **-F** and **-C** options is invoked. When the command or function is invoked, the **COMP_LINE**, **COMP_POINT**, **COMP_KEY**, and **COMP_TYPE** variables are assigned values as described above under **Shell Variables**. If a shell function is being invoked, the **COMP_WORDS** and **COMP_CWORD** variables are also set. When the function or command is invoked, the first argument (**\$1**) is the name of the command whose arguments are being completed, the second argument (**\$2**) is the word being completed, and the third argument (**\$3**) is the word preceding the word being completed on the current command line. No filtering of the generated completions against the word being completed is performed; the function or command has complete freedom in generating the matches.

Any function specified with **-F** is invoked first. The function may use any of the shell facilities, including the **compgen** builtin described below, to generate the matches. It must put the possible completions in the **COMPREPLY** array variable, one per array element.

Next, any command specified with the **-C** option is invoked in an environment equivalent to command substitution. It should print a list of completions, one per line, to the standard output. Backslash may be used to escape a newline, if necessary.

After all of the possible completions are generated, any filter specified with the **-X** option is applied to the list. The filter is a pattern as used for pathname expansion; a **&** in the pattern is replaced with the text of the word being completed. A literal **&** may be escaped with a backslash; the backslash is removed before attempting a match. Any completion that matches the pattern will be removed from the list. A leading **!** negates the pattern; in this case any completion not matching the pattern will be removed. If the **nocase-match** shell option is enabled, the match is performed without regard to the case of alphabetic characters.

Finally, any prefix and suffix specified with the **-P** and **-S** options are added to each member of the completion list, and the result is returned to the readline completion code as the list of possible completions.

If the previously-applied actions do not generate any matches, and the **-o dirnames** option was supplied to **complete** when the compspec was defined, directory name completion is attempted.

If the **-o plusdirs** option was supplied to **complete** when the compspec was defined, directory name

completion is attempted and any matches are added to the results of the other actions.

By default, if a compspec is found, whatever it generates is returned to the completion code as the full set of possible completions. The default **bash** completions are not attempted, and the readline default of file-name completion is disabled. If the **-o bashdefault** option was supplied to **complete** when the compspec was defined, the **bash** default completions are attempted if the compspec generates no matches. If the **-o default** option was supplied to **complete** when the compspec was defined, readline's default completion will be performed if the compspec (and, if attempted, the default **bash** completions) generate no matches.

When a compspec indicates that directory name completion is desired, the programmable completion functions force readline to append a slash to completed names which are symbolic links to directories, subject to the value of the **mark-directories** readline variable, regardless of the setting of the **mark-sym-linked-directories** readline variable.

There is some support for dynamically modifying completions. This is most useful when used in combination with a default completion specified with **complete -D**. It's possible for shell functions executed as completion handlers to indicate that completion should be retried by returning an exit status of 124. If a shell function returns 124, and changes the compspec associated with the command on which completion is being attempted (supplied as the first argument when the function is executed), programmable completion restarts from the beginning, with an attempt to find a new compspec for that command. This allows a set of completions to be built dynamically as completion is attempted, rather than being loaded all at once.

For instance, assuming that there is a library of compspecs, each kept in a file corresponding to the name of the command, the following default completion function would load completions dynamically:

```
_completion_loader()
{
    . "/etc/bash_completion.d/$1.sh" >/dev/null 2>&1 && return 124
}
complete -D -F _completion_loader -o bashdefault -o default
```

HISTORY

When the **-o history** option to the **set** builtin is enabled, the shell provides access to the *command history*, the list of commands previously typed. The value of the **HISTSIZE** variable is used as the number of commands to save in a history list. The text of the last **HISTSIZE** commands (default 500) is saved. The shell stores each command in the history list prior to parameter and variable expansion (see **EXPANSION** above) but after history expansion is performed, subject to the values of the shell variables **HISTIGNORE** and **HISTCONTROL**.

On startup, the history is initialized from the file named by the variable **HISTFILE** (default `~/.bash_history`). The file named by the value of **HISTFILE** is truncated, if necessary, to contain no more than the number of lines specified by the value of **HISTFILESIZE**. If **HISTFILESIZE** is unset, or set to null, a non-numeric value, or a numeric value less than zero, the history file is not truncated. When the history file is read, lines beginning with the history comment character followed immediately by a digit are interpreted as timestamps for the following history line. These timestamps are optionally displayed depending on the value of the **HISTTIMEFORMAT** variable. When a shell with history enabled exits, the last **\$HISTSIZE** lines are copied from the history list to **\$HISTFILE**. If the **histappend** shell option is enabled (see the description of **shopt** under **SHELL BUILTIN COMMANDS** below), the lines are appended to the history file, otherwise the history file is overwritten. If **HISTFILE** is unset, or if the history file is unwritable, the history is not saved. If the **HISTTIMEFORMAT** variable is set, time stamps are written to the history file, marked with the history comment character, so they may be preserved across shell sessions. This uses the history comment character to distinguish timestamps from other history lines. After saving the history, the history file is truncated to contain no more than **HISTFILESIZE** lines. If **HISTFILESIZE** is unset, or set to null, a non-numeric value, or a numeric value less than zero, the history file is not truncated.

The builtin command **fc** (see **SHELL BUILTIN COMMANDS** below) may be used to list or edit and re-execute a portion of the history list. The **history** builtin may be used to display or modify the history list and manipulate the history file. When using command-line editing, search commands are available in each

editing mode that provide access to the history list.

The shell allows control over which commands are saved on the history list. The **HISTCONTROL** and **HISTIGNORE** variables may be set to cause the shell to save only a subset of the commands entered. The **cmdhist** shell option, if enabled, causes the shell to attempt to save each line of a multi-line command in the same history entry, adding semicolons where necessary to preserve syntactic correctness. The **lithist** shell option causes the shell to save the command with embedded newlines instead of semicolons. See the description of the **shopt** builtin below under **SHELL BUILTIN COMMANDS** for information on setting and unsetting shell options.

HISTORY EXPANSION

The shell supports a history expansion feature that is similar to the history expansion in **csh**. This section describes what syntax features are available. This feature is enabled by default for interactive shells, and can be disabled using the **+H** option to the **set** builtin command (see **SHELL BUILTIN COMMANDS** below). Non-interactive shells do not perform history expansion by default.

History expansions introduce words from the history list into the input stream, making it easy to repeat commands, insert the arguments to a previous command into the current input line, or fix errors in previous commands quickly.

History expansion is performed immediately after a complete line is read, before the shell breaks it into words, and is performed on each line individually without taking quoting on previous lines into account. It takes place in two parts. The first is to determine which line from the history list to use during substitution. The second is to select portions of that line for inclusion into the current one. The line selected from the history is the *event*, and the portions of that line that are acted upon are *words*. Various *modifiers* are available to manipulate the selected words. The line is broken into words in the same fashion as when reading input, so that several *metacharacter*-separated words surrounded by quotes are considered one word. History expansions are introduced by the appearance of the history expansion character, which is **!** by default. Only backslash (****) and single quotes can quote the history expansion character, but the history expansion character is also treated as quoted if it immediately precedes the closing double quote in a double-quoted string.

Several characters inhibit history expansion if found immediately following the history expansion character, even if it is unquoted: space, tab, newline, carriage return, and **=**. If the **extglob** shell option is enabled, **(** will also inhibit expansion.

Several shell options settable with the **shopt** builtin may be used to tailor the behavior of history expansion. If the **histverify** shell option is enabled (see the description of the **shopt** builtin below), and **readline** is being used, history substitutions are not immediately passed to the shell parser. Instead, the expanded line is reloaded into the **readline** editing buffer for further modification. If **readline** is being used, and the **histreedit** shell option is enabled, a failed history substitution will be reloaded into the **readline** editing buffer for correction. The **-p** option to the **history** builtin command may be used to see what a history expansion will do before using it. The **-s** option to the **history** builtin may be used to add commands to the end of the history list without actually executing them, so that they are available for subsequent recall.

The shell allows control of the various characters used by the history expansion mechanism (see the description of **histchars** above under **Shell Variables**). The shell uses the history comment character to mark history timestamps when writing the history file.

Event Designators

An event designator is a reference to a command line entry in the history list. Unless the reference is absolute, events are relative to the current position in the history list.

- !** Start a history substitution, except when followed by a **blank**, newline, carriage return, **=** or **(** (when the **extglob** shell option is enabled using the **shopt** builtin).
- !n** Refer to command line *n*.
- !-n** Refer to the current command minus *n*.
- !!** Refer to the previous command. This is a synonym for '**!-1**'.
- !string** Refer to the most recent command preceding the current position in the history list starting with *string*.

!?*string[?]*

Refer to the most recent command preceding the current position in the history list containing *string*. The trailing? may be omitted if *string* is followed immediately by a newline. If *string* is missing, the string from the most recent search is used; it is an error if there is no previous search string.

~*string1***^***string2***^**

Quick substitution. Repeat the previous command, replacing *string1* with *string2*. Equivalent to “`!!:s~string1~string2~`” (see **Modifiers** below).

!#

The entire command line typed so far.

Word Designators

Word designators are used to select desired words from the event. A: separates the event specification from the word designator. It may be omitted if the word designator begins with a ^, \$, *, -, or %. Words are numbered from the beginning of the line, with the first word being denoted by 0 (zero). Words are inserted into the current line separated by single spaces.

0 (zero)

The zeroth word. For the shell, this is the command word.

n

The *n*th word.

^

The first argument. That is, word 1.

\$

The last word. This is usually the last argument, but will expand to the zeroth word if there is only one word in the line.

%

The first word matched by the most recent ‘?*string*?’ search, if the search string begins with a character that is part of a word.

x-y

A range of words; ‘-y’ abbreviates ‘0-y’.

All of the words but the zeroth. This is a synonym for ‘1-\$’. It is not an error to use* if there is just one word in the event; the empty string is returned in that case.

x*

Abbreviates *x-\$*.

x-

Abbreviates *x-\$* like **x***, but omits the last word. If **x** is missing, it defaults to 0.

If a word designator is supplied without an event specification, the previous command is used as the event.

Modifiers

After the optional word designator, there may appear a sequence of one or more of the following modifiers, each preceded by a ‘:’. These modify, or edit, the word or words selected from the history event.

h

Remove a trailing filename component, leaving only the head.

t

Remove all leading filename components, leaving the tail.

r

Remove a trailing suffix of the form .xxx, leaving the basename.

e

Remove all but the trailing suffix.

p

Print the new command but do not execute it.

q

Quote the substituted words, escaping further substitutions.

x

Quote the substituted words as with **q**, but break into words at **blanks** and newlines. The **q** and **x** modifiers are mutually exclusive; the last one supplied is used.

s*old*/*new***/**

Substitute *new* for the first occurrence of *old* in the event line. Any character may be used as the delimiter in place of /. The final delimiter is optional if it is the last character of the event line. The delimiter may be quoted in *old* and *new* with a single backslash. If & appears in *new*, it is replaced by *old*. A single backslash will quote the &. If *old* is null, it is set to the last *old* substituted, or, if no previous history substitutions took place, the last *string* in a !?*string*[?] search. If *new* is null, each matching *old* is deleted.

&

Repeat the previous substitution.

g

Cause changes to be applied over the entire event line. This is used in conjunction with ‘:s’ (e.g., ‘:gs/*old*/*new*/’) or ‘:&’. If used with ‘:s’, any delimiter can be used in place of /, and the final delimiter is optional if it is the last character of the event line. An **a** may be used as a synonym for **g**.

G

Apply the following ‘s’ or ‘&’ modifier once to each word in the event line.

SHELL BUILTIN COMMANDS

Unless otherwise noted, each builtin command documented in this section as accepting options preceded by `-` accepts `--` to signify the end of the options. The `:`, `true`, `false`, and `test/[` builtins do not accept options and do not treat `--` specially. The `exit`, `logout`, `r eturn`, `b reak`, `c ontinue`, `l et`, and `s hift` builtins accept and process arguments beginning with `-` without requiring `--`. Other builtins that accept arguments but are not specified as accepting options interpret arguments beginning with `-` as invalid options and require `--` to prevent this interpretation.

: [arguments]

No effect; the command does nothing beyond expanding *arguments* and performing any specified redirections. The return status is zero.

. filename [arguments]

source filename [arguments]

Read and execute commands from *filename* in the current shell environment and return the exit status of the last command executed from *filename*. If *filename* does not contain a slash, filenames in **PATH** are used to find the directory containing *filename*. The file searched for in **PATH** need not be executable. When **bash** is not in *posix mode*, the current directory is searched if no file is found in **PATH**. If the **sourcepath** option to the **shopt** builtin command is turned off, the **PATH** is not searched. If any *arguments* are supplied, they become the positional parameters when *filename* is executed. Otherwise the positional parameters are unchanged. If the **-T** option is enabled, **source** inherits any trap on **DEBUG**; if it is not, any **DEBUG** trap string is saved and restored around the call to **source**, and **source** unsets the **DEBUG** trap while it executes. If **-T** is not set, and the sourced file changes the **DEBUG** trap, the new value is retained when **source** completes. The return status is the status of the last command exited within the script (0 if no commands are executed), and false if *filename* is not found or cannot be read.

alias [-p] [name[=value] ...]

Alias with no arguments or with the **-p** option prints the list of aliases in the form **alias name=value** on standard output. When arguments are supplied, an alias is defined for each *name* whose *value* is given. A trailing space *invalue* causes the next word to be checked for alias substitution when the alias is expanded. For each *name* in the argument list for which no *value* is supplied, the name and value of the alias is printed. **Alias** returns true unless a *name* is given for which no alias has been defined.

bg [jobspec ...]

Resume each suspended job *jobspec* in the background, as if it had been started with **&**. If *jobspec* is not present, the shell's notion of the *current job* is used. **bg** *jobspec* returns 0 unless run when job control is disabled or, when run with job control enabled, any specified *jobspec* was not found or was started without job control.

bind [-m keymap] [-lpsvPSVX]

bind [-m keymap] [-q function] [-u function] [-r keyseq]

bind [-m keymap] -f filename

bind [-m keymap] -x keyseq:shell-command

bind [-m keymap] keyseq:function-name

bind [-m keymap] keyseq:readline-command

Display current **readline** key and function bindings, bind a key sequence to a **readline** function or macro, or set a **readline** variable. Each non-option argument is a command as it would appear in *.inputrc*, but each binding or command must be passed as a separate argument; e.g., `'"\C-x\C-r":re-read-init-file'`. Options, if supplied, have the following meanings:

-m keymap

Use *keymap* as the keymap to be affected by the subsequent bindings. Acceptable *keymap* names are *emacs*, *emacs-standard*, *emacs-meta*, *emacs-ctlx*, *vi*, *vi-move*, *vi-command*, and *vi-insert*. *vi* is equivalent to *vi-command* (*vi-move* is also a synonym); *emacs* is equivalent to *emacs-standard*.

- l** List the names of all **readline** functions.
- p** Display **readline** function names and bindings in such a way that they can be re-read.
- P** List current **readline** function names and bindings.
- s** Display **readline** key sequences bound to macros and the strings they output in such a way that they can be re-read.
- S** Display **readline** key sequences bound to macros and the strings they output.
- v** Display **readline** variable names and values in such a way that they can be re-read.
- V** List current **readline** variable names and values.
- f** *filename*
Read key bindings from *filename*.
- q** *function*
Query about which keys invoke the named *function*.
- u** *function*
Unbind all keys bound to the named *function*.
- r** *keyseq*
Remove any current binding for *keyseq*.
- x** *keyseq:shell-command*
Cause *shell-command* to be executed whenever *keyseq* is entered. When *shell-command* is executed, the shell sets the **READLINE_LINE** variable to the contents of the **readline** line buffer and the **READLINE_POINT** and **READLINE_MARK** variables to the current location of the insertion point and the saved insertion point (the mark), respectively. If the executed command changes the value of any of **READLINE_LINE**, **READLINE_POINT**, or **READLINE_MARK**, those new values will be reflected in the editing state.
- X** List all key sequences bound to shell commands and the associated commands in a format that can be reused as input.

The return value is 0 unless an unrecognized option is given or an error occurred.

break [*n*]

Exit from within a **for**, **while**, **until**, or **select** loop. If *n* is specified, break *n* levels. *n* must be ≥ 1 . If *n* is greater than the number of enclosing loops, all enclosing loops are exited. The return value is 0 unless *n* is not greater than or equal to 1.

builtin *shell-builtin* [*arguments*]

Execute the specified shell builtin, passing it *arguments*, and return its exit status. This is useful when defining a function whose name is the same as a shell builtin, retaining the functionality of the builtin within the function. The **cd** builtin is commonly redefined this way. The return status is false if *shell-builtin* is not a shell builtin command.

caller [*expr*]

Returns the context of any active subroutine call (a shell function or a script executed with the **.** or **source** builtins). Without *expr*, **caller** displays the line number and source filename of the current subroutine call. If a non-negative integer is supplied as *expr*, **caller** displays the line number, subroutine name, and source file corresponding to that position in the current execution call stack. This extra information may be used, for example, to print a stack trace. The current frame is frame 0. The return value is 0 unless the shell is not executing a subroutine call or *expr* does not correspond to a valid position in the call stack.

cd [−**L**|[−**P** [−**e**]] [−@]] [*dir*]

Change the current directory to *dir*. If *dir* is not supplied, the value of the **HOME** shell variable is the default. Any additional arguments following *dir* are ignored. The variable **CDPATH** defines the search path for the directory containing *dir*: each directory name in **CDPATH** is searched for *dir*. Alternative directory names in **CDPATH** are separated by a colon (:). A null directory name in **CDPATH** is the same as the current directory, i.e., “.”. If *dir* begins with a slash (/), then **CDPATH** is not used. The **-P** option causes **cd** to use the physical directory structure by resolving symbolic links while traversing *dir* and before processing instances of .. in *dir* (see also the **-P**

option to the **set** builtin command); the **-L** option forces symbolic links to be followed by resolving the link after processing instances of .. in *dir*. If.. appears in *dir*, it is processed by removing the immediately previous pathname component from *dir*, back to a slash or the beginning of *dir*. If the **-e** option is supplied with **-P**, and the current working directory cannot be successfully determined after a successful directory change, **cd** will return an unsuccessful status. On systems that support it, the **-@** option presents the extended attributes associated with a file as a directory. An argument of – is converted to **\$OLDPWD** before the directory change is attempted. If a non-empty directory name from **CDPATH** is used, or if – is the first argument, and the directory change is successful, the absolute pathname of the new working directory is written to the standard output. The return value is true if the directory was successfully changed; false otherwise.

command [**-pVv**] *command* [*arg* ...]

Run *command* with *args* suppressing the normal shell function lookup. Only builtin commands or commands found in the **PATH** are executed. If the **-p** option is given, the search for *command* is performed using a default value for **PATH** that is guaranteed to find all of the standard utilities. If either the **-V** or **-v** option is supplied, a description of *command* is printed. The **-v** option causes a single word indicating the command or filename used to invoke *command* to be displayed; the **-V** option produces a more verbose description. If the **-V** or **-v** option is supplied, the exit status is 0 if *command* was found, and 1 if not. If neither option is supplied and an error occurred or *command* cannot be found, the exit status is 127. Otherwise, the exit status of the **command** builtin is the exit status of *command*.

compgen [*option*] [*word*]

Generate possible completion matches for *word* according to the *options*, which may be any option accepted by the **complete** builtin with the exception of **-p** and **-r**, and write the matches to the standard output. When using the **-F** or **-C** options, the various shell variables set by the programmable completion facilities, while available, will not have useful values.

The matches will be generated in the same way as if the programmable completion code had generated them directly from a completion specification with the same flags. If *word* is specified, only those completions matching *word* will be displayed.

The return value is true unless an invalid option is supplied, or no matches were generated.

complete [**-abcdefgjksuv**] [**-o** *comp-option*] [**-DEI**] [**-A** *action*] [**-G** *globpat*] [**-W** *wordlist*]
[**-F** *function*] [**-C** *command*] [**-X** *filterpat*] [**-P** *prefix*] [**-S** *suffix*] *name* [*name* ...]

complete **-pr** [**-DEI**] [*name* ...]

Specify how arguments to each *name* should be completed. If the **-p** option is supplied, or if no options are supplied, existing completion specifications are printed in a way that allows them to be reused as input. The **-r** option removes a completion specification for each *name*, or, if no *names* are supplied, all completion specifications. The **-D** option indicates that other supplied options and actions should apply to the “default” command completion; that is, completion attempted on a command for which no completion has previously been defined. The **-E** option indicates that other supplied options and actions should apply to “empty” command completion; that is, completion attempted on a blank line. The **-I** option indicates that other supplied options and actions should apply to completion on the initial non-assignment word on the line, or after a command delimiter such as ; or |, which is usually command name completion. If multiple options are supplied, the **-D** option takes precedence over **-E**, and both take precedence over **-I**. If any of **-D**, **-E**, or **-I** are supplied, any other *name* arguments are ignored; these completions only apply to the case specified by the option.

The process of applying these completion specifications when word completion is attempted is described above under **Programmable Completion**.

Other options, if specified, have the following meanings. The arguments to the **-G**, **-W**, and **-X** options (and, if necessary, the **-P** and **-S** options) should be quoted to protect them from

expansion before the **complete** builtin is invoked.

-o comp-option

The *comp-option* controls several aspects of the compspec's behavior beyond the simple generation of completions. *comp-option* may be one of:

bashdefault

Perform the rest of the default **bash** completions if the compspec generates no matches.

default Use readline's default filename completion if the compspec generates no matches.

dirnames

Perform directory name completion if the compspec generates no matches.

filenames

Tell readline that the compspec generates filenames, so it can perform any file-name-specific processing (like adding a slash to directory names, quoting special characters, or suppressing trailing spaces). Intended to be used with shell functions.

noquote Tell readline not to quote the completed words if they are filenames (quoting filenames is the default).

nosort Tell readline not to sort the list of possible completions alphabetically.

nospace Tell readline not to append a space (the default) to words completed at the end of the line.

plusdirs After any matches defined by the compspec are generated, directory name completion is attempted and any matches are added to the results of the other actions.

-A action

The *action* may be one of the following to generate a list of possible completions:

alias Alias names. May also be specified as **-a**.

arrayvar

Array variable names.

binding Readline key binding names.

builtin Names of shell builtin commands. May also be specified as **-b**.

command

Command names. May also be specified as **-c**.

directory

Directory names. May also be specified as **-d**.

disabled

Names of disabled shell builtins.

enabled Names of enabled shell builtins.

export Names of exported shell variables. May also be specified as **-e**.

file File names. May also be specified as **-f**.

function

Names of shell functions.

group Group names. May also be specified as **-g**.

helptopic

Help topics as accepted by the **help** builtin.

hostname

Hostnames, as taken from the file specified by the **HOSTFILE** shell variable.

job Job names, if job control is active. May also be specified as **-j**.

keyword

Shell reserved words. May also be specified as **-k**.

running Names of running jobs, if job control is active.

service Service names. May also be specified as **-s**.

setopt Valid arguments for the **-o** option to the **set** builtin.

shopt Shell option names as accepted by the **shopt** builtin.

signal Signal names.

stopped Names of stopped jobs, if job control is active.

user User names. May also be specified as **-u**.

variable Names of all shell variables. May also be specified as **-v**.

-C *command*

command is executed in a subshell environment, and its output is used as the possible completions.

-F *function*

The shell function *function* is executed in the current shell environment. When the function is executed, the first argument (**\$1**) is the name of the command whose arguments are being completed, the second argument (**\$2**) is the word being completed, and the third argument (**\$3**) is the word preceding the word being completed on the current command line. When it finishes, the possible completions are retrieved from the value of the **COMPREPLY** array variable.

-G *globpat*

The pathname expansion pattern *globpat* is expanded to generate the possible completions.

-P *prefix*

prefix is added at the beginning of each possible completion after all other options have been applied.

-S *suffix* *suffix* is appended to each possible completion after all other options have been applied.

-W *wordlist*

The *wordlist* is split using the characters in the **IFS** special variable as delimiters, and each resultant word is expanded. Shell quoting is honored within *wordlist*, in order to provide a mechanism for the words to contain shell metacharacters or characters in the value of **IFS**. The possible completions are the members of the resultant list which match the word being completed.

-X *filterpat*

filterpat is a pattern as used for pathname expansion. It is applied to the list of possible completions generated by the preceding options and arguments, and each completion matching *filterpat* is removed from the list. A leading ! in *filterpat* negates the pattern; in this case, any completion not matching *filterpat* is removed.

The return value is true unless an invalid option is supplied, an option other than **-p** or **-r** is supplied without a *name* argument, an attempt is made to remove a completion specification for a *name* for which no specification exists, or an error occurs adding a completion specification.

compopt [**-o** *option*] [**-DEI**] [**+o** *option*] [*name*]

Modify completion options for each *name* according to the *options*, or for the currently-executing completion if no *names* are supplied. If no *options* are given, display the completion options for each *name* or the current completion. The possible values of *option* are those valid for the **complete** builtin described above. The **-D** option indicates that other supplied options should apply to the “default” command completion; that is, completion attempted on a command for which no completion has previously been defined. The **-E** option indicates that other supplied options should apply to “empty” command completion; that is, completion attempted on a blank line. The **-I** option indicates that other supplied options should apply to completion on the initial non-assignment word on the line, or after a command delimiter such as ; or |, which is usually command name completion.

The return value is true unless an invalid option is supplied, an attempt is made to modify the options for a *name* for which no completion specification exists, or an output error occurs.

continue [*n*]

Resume the next iteration of the enclosing **for**, **while**, **until**, or **select** loop. If *n* is specified, resume at the *n*th enclosing loop. *n* must be ≥ 1 . If *n* is greater than the number of enclosing loops,

the last enclosing loop (the “top-level” loop) is resumed. The return value is 0 unless *n* is not greater than or equal to 1.

declare [**-aAfFgiIlnrux**] [**-p**] [*name[=value]* ...]
typeset [**-aAfFgiIlnrux**] [**-p**] [*name[=value]* ...]

Declare variables and/or give them attributes. If *nonames* are given then display the values of variables. The **-p** option will display the attributes and values of each *name*. When **-p** is used with *name* arguments, additional options, other than **-f** and **-F**, are ignored. When **-p** is supplied without *name* arguments, it will display the attributes and values of all variables having the attributes specified by the additional options. If no other options are supplied with **-p**, **declare** will display the attributes and values of all shell variables. The **-f** option will restrict the display to shell functions. The **-F** option inhibits the display of function definitions; only the function name and attributes are printed. If the **extdebug** shell option is enabled using **shopt**, the source file name and line number where each *name* is defined are displayed as well. The **-F** option implies **-f**. The **-g** option forces variables to be created or modified at the global scope, even when **declare** is executed in a shell function. It is ignored in all other cases. The **-I** option causes local variables to inherit the attributes (except the *nameref* attribute) and value of any existing variable with the same *name* at a surrounding scope. If there is no existing variable, the local variable is initially unset. The following options can be used to restrict output to variables with the specified attribute or to give variables attributes:

- a** Each *name* is an indexed array variable (see **Arrays** above).
- A** Each *name* is an associative array variable (see **Arrays** above).
- f** Use function names only.
- i** The variable is treated as an integer; arithmetic evaluation (see **ARITHMETIC EVALUATION** above) is performed when the variable is assigned a value.
- l** When the variable is assigned a value, all upper-case characters are converted to lower-case. The upper-case attribute is disabled.
- n** Give each *name* the *nameref* attribute, making it a name reference to another variable. That other variable is defined by the value of *name*. All references, assignments, and attribute modifications to *name*, except those using or changing the **-n** attribute itself, are performed on the variable referenced by *name*’s value. The *nameref* attribute cannot be applied to array variables.
- r** Make *names* readonly. These names cannot then be assigned values by subsequent assignment statements or unset.
- t** Give each *name* the *trace* attribute. Traced functions inherit the **DEBUG** and **RETURN** traps from the calling shell. The *trace* attribute has no special meaning for variables.
- u** When the variable is assigned a value, all lower-case characters are converted to upper-case. The lower-case attribute is disabled.
- x** Mark *names* for export to subsequent commands via the environment.

Using ‘+’ instead of ‘-’ turns off the attribute instead, with the exceptions that **+a** and **+A** may not be used to destroy array variables and **+r** will not remove the readonly attribute. When used in a function, **declare** and **typeset** make each *name* local, as with the **local** command, unless the **-g** option is supplied. If a variable name is followed by *=value*, the value of the variable is set to *value*. When using **-a** or **-A** and the compound assignment syntax to create array variables, additional attributes do not take effect until subsequent assignments. The return value is 0 unless an invalid option is encountered, an attempt is made to define a function using **-f** *foo=bar*, an attempt is made to assign a value to a readonly variable, an attempt is made to assign a value to an array variable without using the compound assignment syntax (see **Arrays** above), one of the *names* is not a valid shell variable name, an attempt is made to turn off readonly status for a readonly variable, an attempt is made to turn off array status for an array variable, or an attempt is made to display a non-existent function with **-f**.

dirs [**-clpv**] [**+n**] [**-n**]

Without options, displays the list of currently remembered directories. The default display is on a single line with directory names separated by spaces. Directories are added to the list with the

pushd command; the **popd** command removes entries from the list. The current directory is always the first directory in the stack.

- c** Clears the directory stack by deleting all of the entries.
- l** Produces a listing using full pathnames; the default listing format uses a tilde to denote the home directory.
- p** Print the directory stack with one entry per line.
- v** Print the directory stack with one entry per line, prefixing each entry with its index in the stack.
- +n** Displays the *n*th entry counting from the left of the list shown by **dirs** when invoked without options, starting with zero.
- n** Displays the *n*th entry counting from the right of the list shown by **dirs** when invoked without options, starting with zero.

The return value is 0 unless an invalid option is supplied or *n* indexes beyond the end of the directory stack.

disown [**-ar**] [**-h**] [*jobspec* ... | *pid* ...]

Without options, remove each *jobspec* from the table of active jobs. If *jobspec* is not present, and neither the **-a** nor the **-r** option is supplied, the *current job* is used. If the **-h** option is given, each *jobspec* is not removed from the table, but is marked so that **SIGHUP** is not sent to the job if the shell receives a **SIGHUP**. If no *jobspec* is supplied, the **-a** option means to remove or mark all jobs; the **-r** option without a *jobspec* argument restricts operation to running jobs. The return value is 0 unless a *jobspec* does not specify a valid job.

echo [**-neE**] [*arg* ...]

Output the *args*, separated by spaces, followed by a newline. The return status is 0 unless a write error occurs. If **-n** is specified, the trailing newline is suppressed. If the **-e** option is given, interpretation of the following backslash-escaped characters is enabled. The **-E** option disables the interpretation of these escape characters, even on systems where they are interpreted by default. The **xpg_echo** shell option may be used to dynamically determine whether or not **echo** expands these escape characters by default. **echo** does not interpret -- to mean the end of options. **echo** interprets the following escape sequences:

- \a** alert (bell)
- \b** backspace
- \c** suppress further output
- \e**
- \E** an escape character
- \f** form feed
- \n** new line
- \r** carriage return
- \t** horizontal tab
- \v** vertical tab
- ** backslash
- \0nnn** the eight-bit character whose value is the octal value *nnn* (zero to three octal digits)
- \xHH** the eight-bit character whose value is the hexadecimal value *HH* (one or two hex digits)
- \uHHHH** the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHH* (one to four hex digits)
- \UHHHHHHHHH** the Unicode (ISO/IEC 10646) character whose value is the hexadecimal value *HHHH-HHH* (one to eight hex digits)

enable [**-a**] [**-dnps**] [**-f** *filename*] [*name* ...]

Enable and disable builtin shell commands. Disabling a builtin allows a disk command which has the same name as a shell builtin to be executed without specifying a full pathname, even though the shell normally searches for builtins before disk commands. If **-n** is used, each *name* is disabled; otherwise, *names* are enabled. For example, to use the **test** binary found via the **PATH**

instead of the shell builtin version, run `enable -n test`. The **-f** option means to load the new builtin command *name* from shared object *filename*, on systems that support dynamic loading. The **-d** option will delete a builtin previously loaded with **-f**. If no *name* arguments are given, or if the **-p** option is supplied, a list of shell builtins is printed. With no other option arguments, the list consists of all enabled shell builtins. If **-n** is supplied, only disabled builtins are printed. If **-a** is supplied, the list printed includes all builtins, with an indication of whether or not each is enabled. If **-s** is supplied, the output is restricted to the POSIX *special* builtins. The return value is 0 unless a *name* is not a shell builtin or there is an error loading a new builtin from a shared object.

eval [*arg* ...]

The *args* are read and concatenated together into a single command. This command is then read and executed by the shell, and its exit status is returned as the value of **eval**. If there are no *args*, or only null arguments, **eval** returns 0.

exec [**-cl**] [**-a** *name*] [*command* [*arguments*]]

If *command* is specified, it replaces the shell. No new process is created. The *arguments* become the arguments to *command*. If the **-l** option is supplied, the shell places a dash at the beginning of the zeroth argument passed to *command*. This is what *login(1)* does. The **-c** option causes *command* to be executed with an empty environment. If **-a** is supplied, the shell passes *name* as the zeroth argument to the executed command. If *command* cannot be executed for some reason, a non-interactive shell exits, unless the **execfail** shell option is enabled. In that case, it returns failure. An interactive shell returns failure if the file cannot be executed. A subshell exits unconditionally if **exec** fails. If *command* is not specified, any redirections take effect in the current shell, and the return status is 0. If there is a redirection error, the return status is 1.

exit [*n*] Cause the shell to exit with a status of *n*. If *n* is omitted, the exit status is that of the last command executed. A trap on **EXIT** is executed before the shell terminates.

export [**-fn**] [*name*[=*word*]] ...
export **-p**

The supplied *names* are marked for automatic export to the environment of subsequently executed commands. If the **-f** option is given, the *names* refer to functions. If no *names* are given, or if the **-p** option is supplied, a list of names of all exported variables is printed. The **-n** option causes the export property to be removed from each *name*. If a variable name is followed by **=word**, the value of the variable is set to *word*. **export** returns an exit status of 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **-f** is supplied with a *name* that is not a function.

fc [**-e** *ename*] [**-lnr**] [*first*] [*last*]
fc **-s** [*pat=rep*] [*cmd*]

The first form selects a range of commands from *first* to *last* from the history list and displays or edits and re-executes them. *First* and *last* may be specified as a string (to locate the last command beginning with that string) or as a number (an index into the history list, where a negative number is used as an offset from the current command number). When listing, a *first* or *last* of 0 is equivalent to **-1** and **-0** is equivalent to the current command (usually the **fc** command); otherwise 0 is equivalent to **-1** and **-0** is invalid. If *last* is not specified, it is set to the current command for listing (so that **fc -1 -10** prints the last 10 commands) and to *first* otherwise. If *first* is not specified, it is set to the previous command for editing and **-16** for listing.

The **-n** option suppresses the command numbers when listing. The **-r** option reverses the order of the commands. If the **-l** option is given, the commands are listed on standard output. Otherwise, the editor given by *ename* is invoked on a file containing those commands. If *ename* is not given, the value of the **FCEDIT** variable is used, and the value of **EDITOR** if **FCEDIT** is not set. If neither variable is set, **vi** is used. When editing is complete, the edited commands are echoed and executed.

In the second form, *command* is re-executed after each instance of *pat* is replaced by *rep*.

Command is interpreted the same as *first* above. A useful alias to use with this is `r='fc -s'`, so that typing `r cc` runs the last command beginning with `cc` and typing `r` re-executes the last command.

If the first form is used, the return value is 0 unless an invalid option is encountered or *first* or *last* specify history lines out of range. If the `-e` option is supplied, the return value is the value of the last command executed or failure if an error occurs with the temporary file of commands. If the second form is used, the return status is that of the command re-executed, unless *cmd* does not specify a valid history line, in which case `fc` returns failure.

fg [jobspec]

Resume *jobspec* in the foreground, and make it the current job. If *jobspec* is not present, the shell's notion of the *current job* is used. The return value is that of the command placed into the foreground, or failure if run when job control is disabled or, when run with job control enabled, if *jobspec* does not specify a valid job or *jobspec* specifies a job that was started without job control.

getopts optstring name [arg ...]

getopts is used by shell procedures to parse positional parameters. *optstring* contains the option characters to be recognized; if a character is followed by a colon, the option is expected to have an argument, which should be separated from it by white space. The colon and question mark characters may not be used as option characters. Each time it is invoked, **getopts** places the next option in the shell variable *name*, initializing *name* if it does not exist, and the index of the next argument to be processed into the variable **OPTIND**. **OPTIND** is initialized to 1 each time the shell or a shell script is invoked. When an option requires an argument, **getopts** places that argument into the variable **OPTARG**. The shell does not reset **OPTIND** automatically; it must be manually reset between multiple calls to **getopts** within the same shell invocation if a new set of parameters is to be used.

When the end of options is encountered, **getopts** exits with a return value greater than zero. **OPTIND** is set to the index of the first non-option argument, and *name* is set to `?`.

getopts normally parses the positional parameters, but if more arguments are supplied as *arg* values, **getopts** parses those instead.

getopts can report errors in two ways. If the first character of *optstring* is a colon, *silent* error reporting is used. In normal operation, diagnostic messages are printed when invalid options or missing option arguments are encountered. If the variable **OPTERR** is set to 0, no error messages will be displayed, even if the first character of *optstring* is not a colon.

If an invalid option is seen, **getopts** places `?` into *name* and, if not silent, prints an error message and unsets **OPTARG**. If **getopts** is silent, the option character found is placed in **OPTARG** and no diagnostic message is printed.

If a required argument is not found, and **getopts** is not silent, a question mark (`?`) is placed in *name*, **OPTARG** is unset, and a diagnostic message is printed. If **getopts** is silent, then a colon (`:`) is placed in *name* and **OPTARG** is set to the option character found.

getopts returns true if an option, specified or unspecified, is found. It returns false if the end of options is encountered or an error occurs.

hash [-lr] [-p filename] [-dt] [name]

Each time **hash** is invoked, the full pathname of the command *name* is determined by searching the directories in **\$PATH** and remembered. Any previously-remembered pathname is discarded. If the `-p` option is supplied, no path search is performed, and *filename* is used as the full filename of the command. The `-r` option causes the shell to forget all remembered locations. The `-d` option causes the shell to forget the remembered location of each *name*. If the `-t` option is supplied,

the full pathname to which each *name* corresponds is printed. If multiple *name* arguments are supplied with **-t**, the *name* is printed before the hashed full pathname. The **-I** option causes output to be displayed in a format that may be reused as input. If no arguments are given, or if only **-I** is supplied, information about remembered commands is printed. The return status is true unless a *name* is not found or an invalid option is supplied.

help [**-dms**] [*pattern*]

Display helpful information about builtin commands. If *pattern* is specified, **help** gives detailed help on all commands matching *pattern*; otherwise help for all the builtins and shell control structures is printed.

- d** Display a short description of each *pattern*
- m** Display the description of each *pattern* in a manpage-like format
- s** Display only a short usage synopsis for each *pattern*

The return status is 0 unless no command matches *pattern*.

history [*n*]
history -c
history -d *offset*
history -d *start-end*
history -anrw [*filename*]
history -p *arg* [*arg* ...]
history -s *arg* [*arg* ...]

With no options, display the command history list with line numbers. Lines listed with a * have been modified. An argument of *n* lists only the last *n* lines. If the shell variable **HISTTIMEFORMAT** is set and not null, it is used as a format string for *strftime*(3) to display the time stamp associated with each displayed history entry. No intervening blank is printed between the formatted time stamp and the history line. If *filename* is supplied, it is used as the name of the history file; if not, the value of **HISTFILE** is used. Options, if supplied, have the following meanings:

- c** Clear the history list by deleting all the entries.

-d *offset*

Delete the history entry at position *offset*. If *offset* is negative, it is interpreted as relative to one greater than the last history position, so negative indices count back from the end of the history, and an index of -1 refers to the current **history -d** command.

-d *start-end*

Delete the history entries between positions *start* and *end*, inclusive. Positive and negative values for *start* and *end* are interpreted as described above.

- a** Append the “new” history lines to the history file. These are history lines entered since the beginning of the current **bash** session, but not already appended to the history file.

- n** Read the history lines not already read from the history file into the current history list. These are lines appended to the history file since the beginning of the current **bash** session.

- r** Read the contents of the history file and append them to the current history list.

- w** Write the current history list to the history file, overwriting the history file’s contents.

- p** Perform history substitution on the following *args* and display the result on the standard output. Does not store the results in the history list. Each *arg* must be quoted to disable normal history expansion.

- s** Store the *args* in the history list as a single entry. The last command in the history list is removed before the *args* are added.

If the **HISTTIMEFORMAT** variable is set, the time stamp information associated with each history entry is written to the history file, marked with the history comment character. When the history file is read, lines beginning with the history comment character followed immediately by a digit are interpreted as timestamps for the following history entry. The return value is 0 unless an invalid option is encountered, an error occurs while reading or writing the history file, an invalid *offset* is supplied as an argument to **-d**, or the history expansion supplied as an argument to **-p** fails.

jobs [**-l****n****p****r**] [*jobspec* ...]
jobs **-x** *command* [*args* ...]

The first form lists the active jobs. The options have the following meanings:

- l** List process IDs in addition to the normal information.
- n** Display information only about jobs that have changed status since the user was last notified of their status.
- p** List only the process ID of the job's process group leader.
- r** Display only running jobs.
- s** Display only stopped jobs.

If *jobspec* is given, output is restricted to information about that job. The return status is 0 unless an invalid option is encountered or an invalid *jobspec* is supplied.

If the **-x** option is supplied, **jobs** replaces any *jobspec* found in *command* or *args* with the corresponding process group ID, and executes *command* passing it *args*, returning its exit status.

kill [**-s** *sigspec* | **-n** *signum* | **-sig***sigspec*] [*pid* | *jobspec*] ...
kill **-l**|**-L** [*sigspec* | *exit_status*]

Send the signal named by *sigspec* or *signum* to the processes named by *pid* or *jobspec*. *sigspec* is either a case-insensitive signal name such as **SIGKILL** (with or without the **SIG** prefix) or a signal number; *signum* is a signal number. If *sigspec* is not present, then **SIGTERM** is assumed. An argument of **-l** lists the signal names. If any arguments are supplied when **-l** is given, the names of the signals corresponding to the arguments are listed, and the return status is 0. The *exit_status* argument to **-L** is a number specifying either a signal number or the exit status of a process terminated by a signal. The **-L** option is equivalent to **-l**. **kill** returns true if at least one signal was successfully sent, or false if an error occurs or an invalid option is encountered.

let *arg* [*arg* ...]

Each *arg* is an arithmetic expression to be evaluated (see **ARITHMETIC EVALUATION** above). If the last *arg* evaluates to 0, **let** returns 1; 0 is returned otherwise.

local [*option*] [*name*=*value*] ... | -]

For each argument, a local variable named *name* is created, and assigned *value*. The *option* can be any of the options accepted by **declare**. When **local** is used within a function, it causes the variable *name* to have a visible scope restricted to that function and its children. If *name* is -, the set of shell options is made local to the function in which **local** is invoked: shell options changed using the **set** builtin inside the function are restored to their original values when the function returns. The restore is effected as if a series of **set** commands were executed to restore the values that were in place before the function. With no operands, **local** writes a list of local variables to the standard output. It is an error to use **local** when not within a function. The return status is 0 unless **local** is used outside a function, an invalid *name* is supplied, or *name* is a readonly variable.

logout Exit a login shell.

mapfile [**-d** *delim*] [**-n** *count*] [**-O** *origin*] [**-s** *count*] [**-t**] [**-u** *fd*] [**-C** *callback*] [**-c** *quantum*] [*array*]
readarray [**-d** *delim*] [**-n** *count*] [**-O** *origin*] [**-s** *count*] [**-t**] [**-u** *fd*] [**-C** *callback*] [**-c** *quantum*] [*array*]

Read lines from the standard input into the indexed array variable *array*, or from file descriptor *fd* if the **-u** option is supplied. The variable **MAPFILE** is the default *array*. Options, if supplied, have the following meanings:

- d** The first character of *delim* is used to terminate each input line, rather than newline. If *delim* is the empty string, **mapfile** will terminate a line when it reads a NUL character.
- n** Copy at most *count* lines. If *count* is 0, all lines are copied.
- O** Begin assigning to *array* at index *origin*. The default index is 0.
- s** Discard the first *count* lines read.
- t** Remove a trailing *delim* (default newline) from each line read.
- u** Read lines from file descriptor *fd* instead of the standard input.
- C** Evaluate *callback* each time *quantum* lines are read. The **-c** option specifies *quantum*.
- c** Specify the number of lines read between each call to *callback*.

If **-C** is specified without **-c**, the default quantum is 5000. When *callback* is evaluated, it is supplied the index of the next array element to be assigned and the line to be assigned to that element as additional arguments. *callback* is evaluated after the line is read but before the array element is assigned.

If not supplied with an explicit origin, **mapfile** will clear *array* before assigning to it.

mapfile returns successfully unless an invalid option or option argument is supplied, *array* is invalid or unassignable, or if *array* is not an indexed array.

popd [**-n**] [**+n**] [**-n**]

Removes entries from the directory stack. With no arguments, removes the top directory from the stack, and performs a **cd** to the new top directory. Arguments, if supplied, have the following meanings:

- n** Suppresses the normal change of directory when removing directories from the stack, so that only the stack is manipulated.
- +n** Removes the *n*th entry counting from the left of the list shown by **dirs**, starting with zero. For example: **popd +0** removes the first directory, **popd +1** the second.
- n** Removes the *n*th entry counting from the right of the list shown by **dirs**, starting with zero. For example: **popd -0** removes the last directory, **popd -1** the next to last.

If the **popd** command is successful, a **dirs** is performed as well, and the return status is 0. **popd** returns false if an invalid option is encountered, the directory stack is empty, a non-existent directory stack entry is specified, or the directory change fails.

printf [**-v var**] *format* [*arguments*]

Write the formatted *arguments* to the standard output under the control of the *format*. The **-v** option causes the output to be assigned to the variable *var* rather than being printed to the standard output.

The *format* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences, which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive *argument*. In addition to the standard *printf(1)* format specifications, **printf** interprets the following extensions:

- %b** causes **printf** to expand backslash escape sequences in the corresponding *argument* in the same way as **echo -e**.
- %q** causes **printf** to output the corresponding *argument* in a format that can be reused as shell input.

%(datefmt)T causes **printf** to output the date-time string resulting from using *datefmt* as a format string for *strftime(3)*. The corresponding *argument* is an integer representing the number of seconds since the epoch. Two special argument values may be used: **-1** represents the current time, and **-2** represents the time the shell was invoked. If no argument is specified, conversion behaves as if **-1** had been given. This is an exception to the usual **printf** behavior.

The **%b**, **%q**, and **%T** directives all use the field width and precision arguments from the format specification and write that many bytes from (or use that wide a field for) the expanded argument, which usually contains more characters than the original.

Arguments to non-string format specifiers are treated as C constants, except that a leading plus or minus sign is allowed, and if the leading character is a single or double quote, the value is the ASCII value of the following character.

The *format* is reused as necessary to consume all of the *arguments*. If the *format* requires more *arguments* than are supplied, the extra format specifications behave as if a zero value or null string, as appropriate, had been supplied. The return value is zero on success, non-zero on failure.

pushd [-n] [+n] [-n]
pushd [-n] [dir]

Adds a directory to the top of the directory stack, or rotates the stack, making the new top of the stack the current working directory. With no arguments, **pushd** exchanges the top two directories and returns 0, unless the directory stack is empty. Arguments, if supplied, have the following meanings:

- n Suppresses the normal change of directory when rotating or adding directories to the stack, so that only the stack is manipulated.
- +n Rotates the stack so that the *n*th directory (counting from the left of the list shown by **dirs**, starting with zero) is at the top.
- n Rotates the stack so that the *n*th directory (counting from the right of the list shown by **dirs**, starting with zero) is at the top.
- dir Adds *dir* to the directory stack at the top, making it the new current working directory as if it had been supplied as the argument to the **cd** builtin.

If the **pushd** command is successful, a **dirs** is performed as well. If the first form is used, **pushd** returns 0 unless the **cd** to *dir* fails. With the second form, **pushd** returns 0 unless the directory stack is empty, a non-existent directory stack element is specified, or the directory change to the specified new current directory fails.

pwd [-LP]

Print the absolute pathname of the current working directory. The pathname printed contains no symbolic links if the -P option is supplied or the -o **physical** option to the **set** builtin command is enabled. If the -L option is used, the pathname printed may contain symbolic links. The return status is 0 unless an error occurs while reading the name of the current directory or an invalid option is supplied.

read [-ers] [-a *aname*] [-d *delim*] [-i *text*] [-n *nchars*] [-N *nchars*] [-p *prompt*] [-t *timeout*] [-u *fd*] [*name ...*]

One line is read from the standard input, or from the file descriptor *fd* supplied as an argument to the -u option, split into words as described above under **Word Splitting**, and the first word is assigned to the first *name*, the second word to the second *name*, and so on. If there are more words than names, the remaining words and their intervening delimiters are assigned to the last *name*. If there are fewer words read from the input stream than names, the remaining names are assigned empty values. The characters in **IFS** are used to split the line into words using the same rules the shell uses for expansion (described above under **Word Splitting**). The backslash character (\) may be used to remove any special meaning for the next character read and for line continuation. Options, if supplied, have the following meanings:

-a *aname*

The words are assigned to sequential indices of the array variable *aname*, starting at 0. *aname* is unset before any new values are assigned. Other *name* arguments are ignored.

-d *delim*

The first character of *delim* is used to terminate the input line, rather than newline. If *delim* is the empty string, **read** will terminate a line when it reads a NUL character.

-e If the standard input is coming from a terminal, **readline** (see **READLINE** above) is used to obtain the line. Readline uses the current (or default, if line editing was not previously active) editing settings, but uses Readline's default filename completion.

-i *text* If **readline** is being used to read the line, *text* is placed into the editing buffer before editing begins.

-n *nchars*

read returns after reading *nchars* characters rather than waiting for a complete line of input, but honors a delimiter if fewer than *nchars* characters are read before the delimiter.

-N *nchars*

read returns after reading exactly *nchars* characters rather than waiting for a complete line of input, unless EOF is encountered or **read** times out. Delimiter characters encountered in the input are not treated specially and do not cause **read** to return until *nchars*

characters are read. The result is not split on the characters in **IFS**; the intent is that the variable is assigned exactly the characters read (with the exception of backslash; see the **-r** option below).

-p *prompt*

Display *prompt* on standard error, without a trailing newline, before attempting to read any input. The prompt is displayed only if input is coming from a terminal.

-r

Backslash does not act as an escape character. The backslash is considered to be part of the line. In particular, a backslash-newline pair may not then be used as a line continuation.

-s

Silent mode. If input is coming from a terminal, characters are not echoed.

-t *timeout*

Cause **read** to time out and return failure if a complete line of input (or a specified number of characters) is not read within *timeout* seconds. *timeout* may be a decimal number with a fractional portion following the decimal point. This option is only effective if **read** is reading input from a terminal, pipe, or other special file; it has no effect when reading from regular files. If **read** times out, **read** saves any partial input read into the specified variable *name*. If *timeout* is 0, **read** returns immediately, without trying to read any data. The exit status is 0 if input is available on the specified file descriptor, non-zero otherwise. The exit status is greater than 128 if the timeout is exceeded.

-u *fd*

Read input from file descriptor *fd*.

If no *names* are supplied, the line read, without the ending delimiter but otherwise unmodified, is assigned to the variable **REPLY**. The exit status is zero, unless end-of-file is encountered, **read** times out (in which case the status is greater than 128), a variable assignment error (such as assigning to a readonly variable) occurs, or an invalid file descriptor is supplied as the argument to **-u**.

readonly [**-aAf**] [**-p**] [*name[=word]* ...]

The given *names* are marked readonly; the values of these *names* may not be changed by subsequent assignment. If the **-f** option is supplied, the functions corresponding to the *names* are so marked. The **-a** option restricts the variables to indexed arrays; the **-A** option restricts the variables to associative arrays. If both options are supplied, **-A** takes precedence. If no *name* arguments are given, or if the **-p** option is supplied, a list of all readonly names is printed. The other options may be used to restrict the output to a subset of the set of readonly names. The **-p** option causes output to be displayed in a format that may be reused as input. If a variable name is followed by *=word*, the value of the variable is set to *word*. The return status is 0 unless an invalid option is encountered, one of the *names* is not a valid shell variable name, or **-f** is supplied with a *name* that is not a function.

return [*n*]

Causes a function to stop executing and return the value specified by *n* to its caller. If *n* is omitted, the return status is that of the last command executed in the function body. If **r** **eturn** is executed by a trap handler, the last command used to determine the status is the last command executed before the trap handler. If **r** **eturn** is executed during a **DEBUG** trap, the last command used to determine the status is the last command executed by the trap handler before **return** was invoked. If **return** is used outside a function, but during execution of a script by the **. (source)** command, it causes the shell to stop executing that script and return either *n* or the exit status of the last command executed within the script as the exit status of the script. If *n* is supplied, the return value is its least significant 8 bits. The return status is non-zero if **return** is supplied a non-numeric argument, or is used outside a function and not during execution of a script by **.** or **source**. Any command associated with the **RETURN** trap is executed before execution resumes after the function or script.

set [--**a****b****e****f****h****k****m****n****p****t****u****v****x****B****C****E****H****P****T**] [**-o** *option-name*] [*arg* ...]

set [+**a****b****e****f****h****k****m****n****p****t****u****v****x****B****C****E****H****P****T**] [**+o** *option-name*] [*arg* ...]

Without options, the name and value of each shell variable are displayed in a format that can be reused as input for setting or resetting the currently-set variables. Read-only variables cannot be

reset. In *Posix mode*, only shell variables are listed. The output is sorted according to the current locale. When options are specified, they set or unset shell attributes. Any arguments remaining after option processing are treated as values for the positional parameters and are assigned, in order, to **\$1**, **\$2**, ... **\$n**. Options, if specified, have the following meanings:

- a** Each variable or function that is created or modified is given the **export** attribute and marked for export to the environment of subsequent commands.
- b** Report the status of terminated background jobs immediately, rather than before the next primary prompt. This is effective only when job control is enabled.
- e** Exit immediately if a *pipeline* (which may consist of a single *simple command*), a *list*, or a *compound command* (see **SHELL GRAMMAR** above), exits with a non-zero status. The shell does not exit if the command that fails is part of the command list immediately following a **while** or **until** keyword, part of the test following the **if** or **elif** reserved words, part of any command executed in a **&&** or **||** list except the command following the final **&&** or **||**, any command in a pipeline but the last, or if the command's return value is being inverted with **!**. If a compound command other than a subshell returns a non-zero status because a command failed while **-e** was being ignored, the shell does not exit. A trap on **ERR**, if set, is executed before the shell exits. This option applies to the shell environment and each subshell environment separately (see **COMMAND EXECUTION ENVIRONMENT** above), and may cause subshells to exit before executing all the commands in the subshell.

If a compound command or shell function executes in a context where **-e** is being ignored, none of the commands executed within the compound command or function body will be affected by the **-e** setting, even if **-e** is set and a command returns a failure status. If a compound command or shell function sets **-e** while executing in a context where **-e** is ignored, that setting will not have any effect until the compound command or the command containing the function call completes.

- f** Disable pathname expansion.
- h** Remember the location of commands as they are looked up for execution. This is enabled by default.
- k** All arguments in the form of assignment statements are placed in the environment for a command, not just those that precede the command name.
- m** Monitor mode. Job control is enabled. This option is on by default for interactive shells on systems that support it (see **JOB CONTROL** above). All processes run in a separate process group. When a background job completes, the shell prints a line containing its exit status.
- n** Read commands but do not execute them. This may be used to check a shell script for syntax errors. This is ignored by interactive shells.

-o *option-name*

The *option-name* can be one of the following:

allexport

Same as **-a**.

braceexpand

Same as **-B**.

emacs Use an emacs-style command line editing interface. This is enabled by default when the shell is interactive, unless the shell is started with the **--noediting** option. This also affects the editing interface used for **read -e**.

errexit Same as **-e**.

errtrace Same as **-E**.

functrace

Same as **-T**.

hashall Same as **-h**.

histexpand

Same as **-H**.

history Enable command history, as described above under **HISTORY**. This option is on by default in interactive shells.

ignoreeof

The effect is as if the shell command `IGNOREEOF=10` had been executed (see **Shell Variables** above).

keyword

Same as **-k**.

monitor Same as **-m**.

noclobber

Same as **-C**.

noexec Same as **-n**.

noglob Same as **-f**.

nolog Currently ignored.

notify Same as **-b**.

nounset Same as **-u**.

onecmd Same as **-t**.

physical Same as **-P**.

pipefail If set, the return value of a pipeline is the value of the last (rightmost) command to exit with a non-zero status, or zero if all commands in the pipeline exit successfully. This option is disabled by default.

posix Change the behavior of **bash** where the default operation differs from the POSIX standard to match the standard (*posix mode*). See **SEE ALSO** below for a reference to a document that details how posix mode affects bash's behavior.

privileged

Same as **-p**.

verbose Same as **-v**.

vi Use a vi-style command line editing interface. This also affects the editing interface used for **read -e**.

xtrace Same as **-x**.

If **-o** is supplied with no *option-name*, the values of the current options are printed. If **+o** is supplied with no *option-name*, a series of **set** commands to recreate the current option settings is displayed on the standard output.

-p Turn on *privileged* mode. In this mode, the **\$ENV** and **\$BASH_ENV** files are not processed, shell functions are not inherited from the environment, and the **SHELLOPTS**, **BASHOPTS**, **CDPATH**, and **GLOBIGNORE** variables, if they appear in the environment, are ignored. If the shell is started with the effective user (group) id not equal to the real user (group) id, and the **-p** option is not supplied, these actions are taken and the effective user id is set to the real user id. If the **-p** option is supplied at startup, the effective user id is not reset. Turning this option off causes the effective user and group ids to be set to the real user and group ids.

-t Exit after reading and executing one command.

-u Treat unset variables and parameters other than the special parameters "@" and "*" as an error when performing parameter expansion. If expansion is attempted on an unset variable or parameter, the shell prints an error message, and, if not interactive, exits with a non-zero status.

-v Print shell input lines as they are read.

-x After expanding each *simple command*, **for** command, **case** command, **select** command, or arithmetic **for** command, display the expanded value of **PS4**, followed by the command and its expanded arguments or associated word list.

-B The shell performs brace expansion (see **Brace Expansion** above). This is on by default.

-C If set, **bash** does not overwrite an existing file with the **>**, **>&**, and **<>** redirection operators. This may be overridden when creating output files by using the redirection

- operator `>|` instead of `>`.
- E** If set, any trap on **ERR** is inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **ERR** trap is normally not inherited in such cases.
- H** Enable ! style history substitution. This option is on by default when the shell is interactive.
- P** If set, the shell does not resolve symbolic links when executing commands such as **cd** that change the current working directory. It uses the physical directory structure instead. By default, **bash** follows the logical chain of directories when performing commands which change the current directory.
- T** If set, any traps on **DEBUG** and **RETURN** are inherited by shell functions, command substitutions, and commands executed in a subshell environment. The **DEB UG** and **RETURN** traps are normally not inherited in such cases.
- If no arguments follow this option, then the positional parameters are unset. Otherwise, the positional parameters are set to the *args*, even if some of them begin with a `-`.
- Signal the end of options, cause all remaining *args* to be assigned to the positional parameters. The **-x** and **-v** options are turned off. If there are *no args*, the positional parameters remain unchanged.

The options are off by default unless otherwise noted. Using + rather than – causes these options to be turned off. The options can also be specified as arguments to an invocation of the shell. The current set of options may be found in `$-`. The return status is always true unless an invalid option is encountered.

shift [*n*]

The positional parameters from *n+1* ... are renamed to **\$1** Parameters represented by the numbers `$#` down to `$#-n+1` are unset. *n* must be a non-negative number less than or equal to `$#`. If *n* is 0, no parameters are changed. If *n* is not given, it is assumed to be 1. If *n* is greater than `$#`, the positional parameters are not changed. The return status is greater than zero if *n* is greater than `$#` or less than zero; otherwise 0.

shopt [**-pqsu**] [**-o**] [*optname* ...]

Toggle the values of settings controlling optional shell behavior. The settings can be either those listed below, or, if the **-o** option is used, those available with the **-o** option to the **set** builtin command. With no options, or with the **-p** option, a list of all settable options is displayed, with an indication of whether or not each is set; if *optnames* are supplied, the output is restricted to those options. The **-p** option causes output to be displayed in a form that may be reused as input. Other options have the following meanings:

- s** Enable (set) each *optname*.
- u** Disable (unset) each *optname*.
- q** Suppresses normal output (quiet mode); the return status indicates whether the *optname* is set or unset. If multiple *optname* arguments are given with **-q**, the return status is zero if all *optnames* are enabled; non-zero otherwise.
- o** Restricts the values of *optname* to be those defined for the **-o** option to the **set** builtin.

If either **-s** or **-u** is used with no *optname* arguments, **shopt** shows only those options which are set or unset, respectively. Unless otherwise noted, the **shopt** options are disabled (unset) by default.

The return status when listing options is zero if all *optnames* are enabled, non-zero otherwise. When setting or unsetting options, the return status is zero unless an *optname* is not a valid shell option.

The list of **shopt** options is:

assoc_expand_once

If set, the shell suppresses multiple evaluation of associative array subscripts during arithmetic expression evaluation, while executing builtins that can perform variable assignments, and while executing builtins that perform array dereferencing.

autocd If set, a command name that is the name of a directory is executed as if it were the argument to the **cd** command. This option is only used by interactive shells.

cdable_vars

If set, an argument to the **cd** builtin command that is not a directory is assumed to be the name of a variable whose value is the directory to change to.

cdspell If set, minor errors in the spelling of a directory component in a **cd** command will be corrected. The errors checked for are transposed characters, a missing character, and one character too many. If a correction is found, the corrected filename is printed, and the command proceeds. This option is only used by interactive shells.

checkhash

If set, **bash** checks that a command found in the hash table exists before trying to execute it. If a hashed command no longer exists, a normal path search is performed.

checkjobs

If set, **bash** lists the status of any stopped and running jobs before exiting an interactive shell. If any jobs are running, this causes the exit to be deferred until a second exit is attempted without an intervening command (see **JOB CONTROL** above). The shell always postpones exiting if any jobs are stopped.

checkwinsize

If set, **bash** checks the window size after each external (non-builtin) command and, if necessary, updates the values of **INES** and **COLUMNS**. This option is enabled by default.

cmdhist If set, **bash** attempts to save all lines of a multiple-line command in the same history entry. This allows easy re-editing of multi-line commands. This option is enabled by default, but only has an effect if command history is enabled, as described above under **HISTORY**.

compat31

compat32

compat40

compat41

compat42

compat43

compat44

These control aspects of the shell's compatibility mode (see **SHELL COMPATIBILITY MODE** below).

complete_fullquote

If set, **bash** quotes all shell metacharacters in filenames and directory names when performing completion. If not set, **bash** removes metacharacters such as the dollar sign from the set of characters that will be quoted in completed filenames when these metacharacters appear in shell variable references in words to be completed. This means that dollar signs in variable names that expand to directories will not be quoted; however, any dollar signs appearing in filenames will not be quoted, either. This is active only when bash is using backslashes to quote completed filenames. This variable is set by default, which is the default bash behavior in versions through 4.2.

direxpand

If set, **bash** replaces directory names with the results of word expansion when performing filename completion. This changes the contents of the readline editing buffer. If not set, **bash** attempts to preserve what the user typed.

dirspell If set, **bash** attempts spelling correction on directory names during word completion if the directory name initially supplied does not exist.

dotglob If set, **bash** includes filenames beginning with a ‘.’ in the results of pathname expansion. The filenames “.” and “..” must always be matched explicitly, even if **dotglob** is set.

execfail If set, a non-interactive shell will not exit if it cannot execute the file specified as an argument to the **exec** builtin command. An interactive shell does not exit if **exec** fails.

expand_aliases

If set, aliases are expanded as described above under **ALIASES**. This option is enabled by default for interactive shells.

extdebug

If set at shell invocation, or in a shell startup file, arrange to execute the debugger profile before the shell starts, identical to the **--debugger** option. If set after invocation, behavior intended for use by debuggers is enabled:

1. The **-F** option to the **declare** builtin displays the source file name and line number corresponding to each function name supplied as an argument.
2. If the command run by the **DEBUG** trap returns a non-zero value, the next command is skipped and not executed.
3. If the command run by the **DEBUG** trap returns a value of 2, and the shell is executing in a subroutine (a shell function or a shell script executed by the **.** or **source** builtins), the shell simulates a call to **return**.
4. **BASH_ARGC** and **BASH_ARGV** are updated as described in their descriptions above.
5. Function tracing is enabled: command substitution, shell functions, and subshells invoked with (*command*) inherit the **DEBUG** and **RETURN** traps.
6. Error tracing is enabled: command substitution, shell functions, and subshells invoked with (*command*) inherit the **ERR** trap.

extglob If set, the extended pattern matching features described above under **Pathname Expansion** are enabled.

extquote

If set, `$'string'` and `$$"string"` quoting is performed within `$(parameter)` expansions enclosed in double quotes. This option is enabled by default.

failglob If set, patterns which fail to match filenames during pathname expansion result in an expansion error.

force_ignore

If set, the suffixes specified by the **IGNORE** shell variable cause words to be ignored when performing word completion even if the ignored words are the only possible completions. See **SHELL VARIABLES** above for a description of **IGNORE**. This option is enabled by default.

globasciiranges

If set, range expressions used in pattern matching bracket expressions (see **Pattern Matching** above) behave as if in the traditional C locale when performing comparisons. That is, the current locale's collating sequence is not taken into account, so **b** will not collate between **A** and **B**, and upper-case and lower-case ASCII characters will collate together.

globstar If set, the pattern ****** used in a pathname expansion context will match all files and zero or more directories and subdirectories. If the pattern is followed by a **/**, only directories and subdirectories match.

gnu_errfmt

If set, shell error messages are written in the standard GNU error message format.

histappend

If set, the history list is appended to the file named by the value of the **HISTFILE** variable when the shell exits, rather than overwriting the file.

histreedit

If set, and **readline** is being used, a user is given the opportunity to re-edit a failed history substitution.

histverify

If set, and **readline** is being used, the results of history substitution are not immediately passed to the shell parser. Instead, the resulting line is loaded into the **readline** editing buffer, allowing further modification.

hostcomplete

If set, and **readline** is being used, **bash** will attempt to perform hostname completion when a word containing a @ is being completed (see **Completing under READLINE** above). This is enabled by default.

huponexit

If set, **bash** will send **SIGHUP** to all jobs when an interactive login shell exits.

inherit_errexit

If set, command substitution inherits the value of the **errexit** option, instead of unsetting it in the subshell environment. This option is enabled when *posix mode* is enabled.

interactive_comments

If set, allow a word beginning with # to cause that word and all remaining characters on that line to be ignored in an interactive shell (see **COMMENTS** above). This option is enabled by default.

lastpipe If set, and job control is not active, the shell runs the last command of a pipeline not executed in the background in the current shell environment.

lithist If set, and the **cmdhist** option is enabled, multi-line commands are saved to the history with embedded newlines rather than using semicolon separators where possible.

localvar_inherit

If set, local variables inherit the value and attributes of a variable of the same name that exists at a previous scope before any new value is assigned. The **nameref** attribute is not inherited.

localvar_unset

If set, calling **unset** on local variables in previous function scopes marks them so subsequent lookups find them unset until that function returns. This is identical to the behavior of unsetting local variables at the current function scope.

login_shell

The shell sets this option if it is started as a login shell (see **INVOCATION** above). The value may not be changed.

mailwarn

If set, and a file that **bash** is checking for mail has been accessed since the last time it was checked, the message “The mail in *mailfile* has been read” is displayed.

no_empty_cmd_completion

If set, and **readline** is being used, **bash** will not attempt to search the **PATH** for possible completions when completion is attempted on an empty line.

nocaseglob

If set, **bash** matches filenames in a case-insensitive fashion when performing pathname expansion (see **Pathname Expansion** above).

nocasematch

If set, **bash** matches patterns in a case-insensitive fashion when performing matching while executing **case** or [[conditional commands, when performing pattern substitution word expansions, or when filtering possible completions as part of programmable completion.

nullglob

If set, **bash** allows patterns which match no files (see **Pathname Expansion** above) to expand to a null string, rather than themselves.

progcomp

If set, the programmable completion facilities (see **Programmable Completion** above) are enabled. This option is enabled by default.

progcomp_alias

If set, and programmable completion is enabled, **bash** treats a command name that doesn't have any completions as a possible alias and attempts alias expansion. If it has an alias, **bash** attempts programmable completion using the command word resulting from the expanded alias.

promptvars

If set, prompt strings undergo parameter expansion, command substitution, arithmetic expansion, and quote removal after being expanded as described in **PROMPTING** above. This option is enabled by default.

restricted_shell

The shell sets this option if it is started in restricted mode (see **RESTRICTED SHELL** below). The value may not be changed. This is not reset when the startup files are executed, allowing the startup files to discover whether or not a shell is restricted.

shift_verbose

If set, the **shift** builtin prints an error message when the shift count exceeds the number of positional parameters.

sourcepath

If set, the **source** (.) builtin uses the value of **PATH** to find the directory containing the file supplied as an argument. This option is enabled by default.

xpg_echo

If set, the **echo** builtin expands backslash-escape sequences by default.

suspend [-f]

Suspend the execution of this shell until it receives a **SIGCONT** signal. A login shell cannot be suspended; the **-f** option can be used to override this and force the suspension. The return status is 0 unless the shell is a login shell and **-f** is not supplied, or if job control is not enabled.

test expr

[*expr*] Return a status of 0 (true) or 1 (false) depending on the evaluation of the conditional expression *expr*. Each operator and operand must be a separate argument. Expressions are composed of the primaries described above under **CONDITIONAL EXPRESSIONS**. **test** does not accept an **y** options, nor does it accept and ignore an argument of **--** as signifying the end of options.

Expressions may be combined using the following operators, listed in decreasing order of precedence. The evaluation depends on the number of arguments; see below. Operator precedence is used when there are five or more arguments.

! *expr* True if *expr* is false.

(*expr*) Returns the value of *expr*. This may be used to override the normal precedence of operators.

expr1* -a *expr2

True if both *expr1* and *expr2* are true.

expr1* -o *expr2

True if either *expr1* or *expr2* is true.

test and [evaluate conditional expressions using a set of rules based on the number of arguments.

0 arguments

The expression is false.

1 argument

The expression is true if and only if the argument is not null.

2 arguments

If the first argument is **!**, the expression is true if and only if the second argument is null. If the first argument is one of the unary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the expression is true if the unary test is true. If the first argument is not a valid unary conditional operator, the expression is false.

3 arguments

The following conditions are applied in the order listed. If the second argument is one of the binary conditional operators listed above under **CONDITIONAL EXPRESSIONS**, the result of the expression is the result of the binary test using the first and third arguments as operands. The **-a** and **-o** operators are considered binary operators when there are three arguments. If the first argument is **!**, the value is the negation of the two-argument test using the second and third arguments. If the first argument is exactly **(** and the third argument is exactly **)**, the result is the one-argument test of the second argument. Otherwise, the expression is false.

4 arguments

If the first argument is **!**, the result is the negation of the three-argument expression composed of the remaining arguments. Otherwise, the expression is parsed and evaluated according to precedence using the rules listed above.

5 or more arguments

The expression is parsed and evaluated according to precedence using the rules listed above.

When used with **test** or **[**, the **<** and **>** operators sort lexicographically using ASCII ordering.

times Print the accumulated user and system times for the shell and for processes run from the shell. The return status is 0.

trap [-lp] [[arg] sigspec ...]

The command *arg* is to be read and executed when the shell receives signal(s) *sigspec*. If *arg* is absent (and there is a single *sigspec*) or **-**, each specified signal is reset to its original disposition (the value it had upon entrance to the shell). If *arg* is the null string the signal specified by each *sigspec* is ignored by the shell and by the commands it invokes. If *arg* is not present and **-p** has been supplied, then the trap commands associated with each *sigspec* are displayed. If no arguments are supplied or if only **-p** is given, **trap** prints the list of commands associated with each signal. The **-l** option causes the shell to print a list of signal names and their corresponding numbers. Each *sigspec* is either a signal name defined in *<signal.h>*, or a signal number. Signal names are case insensitive and the **SIG** prefix is optional.

If a *sigspec* is **EXIT** (0) the command *arg* is executed on exit from the shell. If a *sigspec* is **DEBUG**, the command *arg* is executed before every *simple command*, *for* command, *case* command, *select* command, every arithmetic *for* command, and before the first command executes in a shell function (see **SHELL GRAMMAR** above). Refer to the description of the **extdeb ug** option to the **shopt** builtin for details of its effect on the **DEBUG** trap. If a *sigspec* is **RETURN**, the command *arg* is executed each time a shell function or a script executed with the **.** or **source** builtins finishes executing.

If a *sigspec* is **ERR**, the command *arg* is executed whenever a pipeline (which may consist of a single simple command), a list, or a compound command returns a non-zero exit status, subject to the following conditions. The **ERR** trap is not executed if the failed command is part of the command list immediately following a **while** or **until** keyword, part of the test in an **if** statement, part of a command executed in a **&&** or **||** list except the command following the final **&&** or **||**, any command in a pipeline but the last, or if the command's return value is being inverted using **!**. These are the same conditions obeyed by the **errexit** (**-e**) option.

Signals ignored upon entry to the shell cannot be trapped or reset. Trapped signals that are not being ignored are reset to their original values in a subshell or subshell environment when one is created. The return status is false if any *sigspec* is invalid; otherwise **trap** returns true.

type [**-a****f****t****P**] *name* [*name* ...]

With no options, indicate how each *name* would be interpreted if used as a command name. If the **-t** option is used, **type** prints a string which is one of *alias*, *keyword*, *function*, *builtin*, or *file* if *name* is an alias, shell reserved word, function, builtin, or disk file, respectively. If the *name* is not found, then nothing is printed, and an exit status of false is returned. If the **-p** option is used, **type** either returns the name of the disk file that would be executed if *name* were specified as a command name, or nothing if **type -t name** would not return *file*. The **-P** option forces a **P ATH** search for each *name*, even if **type -t name** would not return *file*. If a command is hashed, **-p** and **-P** print the hashed value, which is not necessarily the file that appears first in **PATH**. If the **-a** option is used, **type** prints all of the places that contain an executable named *name*. This includes aliases and functions, if and only if the **-p** option is not also used. The table of hashed commands is not consulted when using **-a**. The **-f** option suppresses shell function lookup, as with the **command** builtin. **type** returns true if all of the arguments are found, false if any are not found.

ulimit [**-HS**] **-a**
ulimit [**-HS**] [**-bcdefiklmnpqrstuvxPRT** [*limit*]]

Provides control over the resources available to the shell and to processes started by it, on systems that allow such control. The **-H** and **-S** options specify that the hard or soft limit is set for the given resource. A hard limit cannot be increased by a non-root user once it is set; a soft limit may be increased up to the value of the hard limit. If neither **-H** nor **-S** is specified, both the soft and hard limits are set. The value of *limit* can be a number in the unit specified for the resource or one of the special values **hard**, **soft**, or **unlimited**, which stand for the current hard limit, the current soft limit, and no limit, respectively. If *limit* is omitted, the current value of the soft limit of the resource is printed, unless the **-H** option is given. When more than one resource is specified, the limit name and unit, if appropriate, are printed before the value. Other options are interpreted as follows:

- a** All current limits are reported; no limits are set
- b** The maximum socket buffer size
- c** The maximum size of core files created
- d** The maximum size of a process's data segment
- e** The maximum scheduling priority ("nice")
- f** The maximum size of files written by the shell and its children
- i** The maximum number of pending signals
- k** The maximum number of kqueues that may be allocated
- l** The maximum size that may be locked into memory
- m** The maximum resident set size (many systems do not honor this limit)
- n** The maximum number of open file descriptors (most systems do not allow this value to be set)
- p** The pipe size in 512-byte blocks (this may not be set)
- q** The maximum number of bytes in POSIX message queues
- r** The maximum real-time scheduling priority
- s** The maximum stack size
- t** The maximum amount of cpu time in seconds
- u** The maximum number of processes available to a single user
- v** The maximum amount of virtual memory available to the shell and, on some systems, to its children
- x** The maximum number of file locks
- P** The maximum number of pseudoterminals
- R** The maximum time a real-time process can run before blocking, in microseconds
- T** The maximum number of threads

If *limit* is given, and the **-a** option is not used, *limit* is the new value of the specified resource. If no option is given, then **-f** is assumed. Values are in 1024-byte increments, except for **-t**, which is in seconds; **-R**, which is in microseconds; **-p**, which is in units of 512-byte blocks; **-P**, **-T**, **-b**, **-k**, **-n**, and **-u**, which are unscaled values; and, when in posix mode, **-c** and **-f**, which are in 512-byte increments. The return status is 0 unless an invalid option or argument is supplied, or an error occurs while setting a new limit.

umask [**-p**] [**-S**] [*mode*]

The user file-creation mask is set to *mode*. If *mode* begins with a digit, it is interpreted as an octal number; otherwise it is interpreted as a symbolic mode mask similar to that accepted by *chmod(1)*. If *mode* is omitted, the current value of the mask is printed. The **-S** option causes the mask to be printed in symbolic form; the default output is an octal number. If the **-p** option is supplied, and *mode* is omitted, the output is in a form that may be reused as input. The return status is 0 if the mode was successfully changed or if no *mode* argument was supplied, and false otherwise.

unalias [**-a**] [*name* ...]

Remove each *name* from the list of defined aliases. If **-a** is supplied, all alias definitions are removed. The return value is true unless a supplied *name* is not a defined alias.

unset [**-fv**] [**-n**] [*name* ...]

For each *name*, remove the corresponding variable or function. If the **-v** option is given, each *name* refers to a shell variable, and that variable is removed. Read-only variables may not be unset. If **-f** is specified, each *name* refers to a shell function, and the function definition is removed. If the **-n** option is supplied, and *name* is a variable with the *nameref* attribute, *name* will be unset rather than the variable it references. **-n** has no effect if the **-f** option is supplied. If no options are supplied, each *name* refers to a variable; if there is no variable by that name, a function with that name, if any, is unset. Each unset variable or function is removed from the environment passed to subsequent commands. If any of **BASH_ALIASES**, **BASH_ARGV0**, **BASH_CMDS**, **BASH_COMMAND**, **BASH_SUBSHELL**, **BASHPID**, **COMP_WORDBREAKS**, **DIRSTACK**, **EPOCHREALTIME**, **EPOCHSECONDS**, **FUNCNAME**, **GROUPS**, **HISTCMD**, **LINENO**, **RANDOM**, **SECONDS**, or **RANDOM** are unset, they lose their special properties, even if they are subsequently reset. The exit status is true unless a *name* is readonly.

wait [**-fn**] [**-p** *varname*] [*id* ...]

Wait for each specified child process and return its termination status. Each *id* may be a process ID or a job specification; if a job spec is given, all processes in that job's pipeline are waited for. If *id* is not given, **wait** waits for all running background jobs and the last-executed process substitution, if its process id is the same as **!\$!**, and the return status is zero. If the **-n** option is supplied, **wait** waits for a single job from the list of *ids* or, if no *ids* are supplied, any job, to complete and returns its exit status. If none of the supplied arguments is a child of the shell, or if no arguments are supplied and the shell has no unwaited-for children, the exit status is 127. If the **-p** option is supplied, the process or job identifier of the job for which the exit status is returned is assigned to the variable *varname* named by the option argument. The variable will be unset initially, before any assignment. This is useful only when the **-n** option is supplied. Supplying the **-f** option, when job control is enabled, forces **wait** to wait for *id* to terminate before returning its status, instead of returning when it changes status. If *id* specifies a non-existent process or job, the return status is 127. Otherwise, the return status is the exit status of the last process or job waited for.

SHELL COMPATIBILITY MODE

Bash-4.0 introduced the concept of a ‘shell compatibility level’, specified as a set of options to the shopt builtin **compat31**, **compat32**, **compat40**, **compat41**, and so on). There is only one current compatibility level -- each option is mutually exclusive. The compatibility level is intended to allow users to select behavior from previous versions that is incompatible with newer versions while they migrate scripts to use current features and behavior. It’s intended to be a temporary solution.

This section does not mention behavior that is standard for a particular version (e.g., setting **compat32** means that quoting the rhs of the regexp matching operator quotes special regexp characters in the word, which is default behavior in bash-3.2 and above).

If a user enables, say, **compat32**, it may affect the behavior of other compatibility levels up to and including the current compatibility level. The idea is that each compatibility level controls behavior that changed in that version of **bash**, but that behavior may have been present in earlier versions. For instance, the change to use locale-based comparisons with the `[[` command came in bash-4.1, and earlier versions used ASCII-based comparisons, so enabling **compat32** will enable ASCII-based comparisons as well. That granularity may not be sufficient for all uses, and as a result users should employ compatibility levels carefully. Read the documentation for a particular feature to find out the current behavior.

Bash-4.3 introduced a new shell variable: **BASH_COMPAT**. The value assigned to this variable (a decimal version number like 4.2, or an integer corresponding to the **compatNN** option, like 42) determines the compatibility level.

Starting with bash-4.4, Bash has begun deprecating older compatibility levels. Eventually, the options will be removed in favor of **BASH_COMPAT**.

Bash-5.0 is the final version for which there will be an individual shopt option for the previous version. Users should use **BASH_COMPAT** on bash-5.0 and later versions.

The following table describes the behavior changes controlled by each compatibility level setting. The **compatNN** tag is used as shorthand for setting the compatibility level to *NN* using one of the following mechanisms. For versions prior to bash-5.0, the compatibility level may be set using the corresponding **compatNN** shopt option. For bash-4.3 and later versions, the **BASH_COMPAT** variable is preferred, and it is required for bash-5.1 and later versions.

compat31

- quoting the rhs of the `[[` command's regexp matching operator (`=~`) has no special effect

compat32

- interrupting a command list such as "a ; b ; c" causes the execution of the next command in the list (in bash-4.0 and later versions, the shell acts as if it received the interrupt, so interrupting one command in a list aborts the execution of the entire list)

compat40

- the `<` and `>` operators to the `[[` command do not consider the current locale when comparing strings; they use ASCII ordering. Bash versions prior to bash-4.1 use ASCII collation and *strcmp(3)*; bash-4.1 and later use the current locale's collation sequence and *strcoll(3)*.

compat41

- in *posix* mode, **time** may be followed by options and still be recognized as a reserved word (this is POSIX interpretation 267)
- in *posix* mode, the parser requires that an even number of single quotes occur in the *word* portion of a double-quoted parameter expansion and treats them specially, so that characters within the single quotes are considered quoted (this is POSIX interpretation 221)

compat42

- the replacement string in double-quoted pattern substitution does not undergo quote removal, as it does in versions after bash-4.2
- in *posix* mode, single quotes are considered special when expanding the *word* portion of a double-quoted parameter expansion and can be used to quote a closing brace or other special character (this is part of POSIX interpretation 221); in later versions, single quotes are not special within double-quoted word expansions

compat43

- the shell does not print a warning message if an attempt is made to use a quoted compound assignment as an argument to declare (declare -a foo='(1 2)'). Later versions warn that this usage is deprecated
- word expansion errors are considered non-fatal errors that cause the current command to fail, even in *posix* mode (the default behavior is to make them fatal errors that cause the shell to exit)

- when executing a shell function, the loop state (while/until/etc.) is not reset, so **break** or **continue** in that function will break or continue loops in the calling context. Bash-4.4 and later reset the loop state to prevent this

compat44

- the shell sets up the values used by **BASH_ARGV** and **BASH_ARGC** so they can expand to the shell's positional parameters even if extended debugging mode is not enabled
- a subshell inherits loops from its parent context, so **break** or **continue** will cause the subshell to exit. Bash-5.0 and later reset the loop state to prevent the exit
- variable assignments preceding builtins like **export** and **readonly** that set attributes continue to affect variables with the same name in the calling environment even if the shell is not in posix mode

compat50

- Bash-5.1 changed the way **\$RANDOM** is generated to introduce slightly more randomness. If the shell compatibility level is set to 50 or lower, it reverts to the method from bash-5.0 and previous versions, so seeding the random number generator by assigning a value to **RANDOM** will produce the same sequence as in bash-5.0
- If the command hash table is empty, bash versions prior to bash-5.1 printed an informational message to that effect, even when producing output that can be reused as input. Bash-5.1 suppresses that message when the **-I** option is supplied.

RESTRICTED SHELL

If **bash** is started with the name **rbash**, or the **-r** option is supplied at invocation, the shell becomes restricted. A restricted shell is used to set up an environment more controlled than the standard shell. It behaves identically to **bash** with the exception that the following are disallowed or not performed:

- changing directories with **cd**
- setting or unsetting the values of **SHELL**, **PATH**, **HISTFILE**, **ENV**, or **BASH_ENV**
- specifying command names containing /
- specifying a filename containing a / as an argument to the . builtin command
- specifying a filename containing a slash as an argument to the **history** builtin command
- specifying a filename containing a slash as an argument to the **-p** option to the **hash** builtin command
- importing function definitions from the shell environment at startup
- parsing the value of **SHELLOPTS** from the shell environment at startup
- redirecting output using the >, >|, <>, >&, &>, and >> redirection operators
- using the **exec** builtin command to replace the shell with another command
- adding or deleting builtin commands with the **-f** and **-d** options to the **enable** builtin command
- using the **enable** builtin command to enable disabled shell builtins
- specifying the **-p** option to the **command** builtin command
- turning off restricted mode with **set +r** or **set +o restricted**.

These restrictions are enforced after any startup files are read.

When a command that is found to be a shell script is executed (see **COMMAND EXECUTION** above), **rbash** turns off any restrictions in the shell spawned to execute the script.

SEE ALSO

Bash Reference Manual, Brian Fox and Chet Ramey
The Gnu Readline Library, Brian Fox and Chet Ramey

The Gnu History Library, Brian Fox and Chet Ramey
Portable Operating System Interface (POSIX) Part 2: Shell and Utilities, IEEE --
<http://pubs.opengroup.org/onlinepubs/9699919799/>
<http://tiswww.case.edu/~chet/bash/POSIX> -- a description of posix mode
sh(1), *ksh(1)*, *csh(1)*
emacs(1), *vi(1)*
readline(3)

FILES

/bin/bash
 The **bash** executable
/etc/profile
 The systemwide initialization file, executed for login shells
/etc/bash.bashrc
 The systemwide per-interactive-shell startup file
/etc/bash.bash.logout
 The systemwide login shell cleanup file, executed when a login shell exits
~/.bash_profile
 The personal initialization file, executed for login shells
~/.bashrc
 The individual per-interactive-shell startup file
~/.bash_logout
 The individual login shell cleanup file, executed when a login shell exits
~/.inputrc
 Individual *readline* initialization file

AUTHORS

Brian Fox, Free Software Foundation
bfox@gnu.org

Chet Ramey, Case Western Reserve University
chet.ramey@case.edu

BUG REPORTS

If you find a bug in **bash**, you should report it. But first, you should make sure that it really is a bug, and that it appears in the latest version of **bash**. The latest version is always available from <ftp://ftp.gnu.org/pub/gnu/bash/>.

Once you have determined that a bug actually exists, use the *bashbug* command to submit a bug report. If you have a fix, you are encouraged to mail that as well! Suggestions and ‘philosophical’ bug reports may be mailed to *bug-bash@gnu.org* or posted to the Usenet newsgroup **gnu.bash.bug**.

ALL bug reports should include:

- The version number of **bash**
- The hardware and operating system
- The compiler used to compile
- A description of the bug behaviour
- A short script or ‘recipe’ which exercises the bug

bashbug inserts the first three items automatically into the template it provides for filing a bug report.

Comments and bug reports concerning this manual page should be directed to *chet.ramey@case.edu*.

BUGS

It’s too big and too slow.

There are some subtle differences between **bash** and traditional versions of **sh**, mostly because of the **POSIX** specification.

Aliases are confusing in some uses.

Shell builtin commands and functions are not stoppable/restartable.

Compound commands and command sequences of the form ‘a ; b ; c’ are not handled gracefully when process suspension is attempted. When a process is stopped, the shell immediately executes the next command in the sequence. It suffices to place the sequence of commands between parentheses to force it into a subshell, which may be stopped as a unit.

Array variables may not (yet) be exported.

There may be only one active coprocess at a time.

bg(1) - Linux man page

Name

bash, :, .., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopt, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see **bash(1)**

Bash Builtin Commands

See Also

bash(1), **sh(1)**

NAME

blkid – locate/print block device attributes

SYNOPSIS

blkid **--label** *label* | **--uuid** *uuid*

blkid [**--no-encoding** **--garbage-collect** **--list-one** **--cache-file** *file*] [**--output** *format*]
[**--match-tag** *tag*] [**--match-token** *NAME=value*] [*device...*]

blkid **--probe** [**--offset** *offset*] [**--output** *format*] [**--size** *size*] [**--match-tag** *tag*] [**--match-types** *list*]
[**--usages** *list*] [**--no-part-details**] *device...*

blkid **--info** [**--output** *format*] [**--match-tag** *tag*] *device...*

DESCRIPTION

The **blkid** program is the command-line interface to working with the **libblkid**(3) library. It can determine the type of content (e.g., filesystem or swap) that a block device holds, and also the attributes (tokens, NAME=value pairs) from the content metadata (e.g., LABEL or UUID fields).

It is recommended to use lsblk(8) command to get information about block devices, or lsblk --fs to get an overview of filesystems, or findmnt(8) to search in already mounted filesystems.

lsblk(8) provides more information, better control on output formatting, easy to use in scripts and it does not require root permissions to get actual information. **blkid** reads information directly from devices and for non-root users it returns cached unverified information. **blkid** is mostly designed for system services and to test **libblkid**(3) functionality.

When *device* is specified, tokens from only this device are displayed. It is possible to specify multiple *device* arguments on the command line. If none is given, all partitions or unpartitioned devices which appear in */proc/partitions* are shown, if they are recognized.

blkid has two main forms of operation: either searching for a device with a specific NAME=value pair, or displaying NAME=value pairs for one or more specified devices.

For security reasons **blkid** silently ignores all devices where the probing result is ambivalent (multiple colliding filesystems are detected). The low-level probing mode (**-p**) provides more information and extra exit status in this case. It's recommended to use **wipefs(8)** to get a detailed overview and to erase obsolete stuff (magic strings) from the device.

OPTIONS

The *size* and *offset* arguments may be followed by the multiplicative suffixes like KiB (=1024), MiB (=1024*1024), and so on for GiB, TiB, PiB, EiB, ZiB and YiB (the "iB" is optional, e.g., "K" has the same meaning as "KiB"), or the suffixes KB (=1000), MB (=1000*1000), and so on for GB, TB, PB, EB, ZB and YB.

-c, --cache-file *cachefile*

Read from *cachefile* instead of reading from the default cache file (see the **CONFIGURATION FILE** section for more details). If you want to start with a clean cache (i.e., don't report devices previously scanned but not necessarily available at this time), specify */dev/null*.

-d, --no-encoding

Don't encode non-printing characters. The non-printing characters are encoded by ^ and M- notation by default. Note that the **--output udev** output format uses a different encoding which cannot be disabled.

-D, --no-part-details

Don't print information (PART_ENTRY_* tags) from partition table in low-level probing mode.

-g, --garbage-collect

Perform a garbage collection pass on the blkid cache to remove devices which no longer exist.

-H, --hint *setting*

Set probing hint. The hints are an optional way to force probing functions to check, for example, another location. The currently supported is "session_offset=number" to set session offset on multi-session UDF.

-i, --info

Display information about I/O Limits (aka I/O topology). The 'export' output format is automatically enabled. This option can be used together with the **--probe** option.

-k, --list-filesystems

List all known filesystems and RAIDs and exit.

-l, --list-one

Look up only one device that matches the search parameter specified with the **--match-token** option. If there are multiple devices that match the specified search parameter, then the device with the highest priority is returned, and/or the first device found at a given priority (but see below note about udev). Device types in order of decreasing priority are: Device Mapper, EVMS, LVM, MD, and finally regular block devices. If this option is not specified, **blkid** will print all of the devices that match the search parameter.

This option forces **blkid** to use udev when used for LABEL or UUID tokens in **--match-token**. The goal is to provide output consistent with other utils (like **mount(8)**, etc.) on systems where the same tag is used for multiple devices.

-L, --label *label*

Look up the device that uses this filesystem *label*; this is equal to **--list-one --output device --match-token LABEL=label**. This lookup method is able to reliably use /dev/disk/by-label udev symlinks (dependent on a setting in */etc/blkid.conf*). Avoid using the symlinks directly; it is not reliable to use the symlinks without verification. The **--label** option works on systems with and without udev.

Unfortunately, the original **blkid(8)** from e2fsprogs uses the **-L** option as a synonym for **-o list**. For better portability, use **-l -o device -t LABEL=label** and **-o list** in your scripts rather than the **-L** option.

-n, --match-types *list*

Restrict the probing functions to the specified (comma-separated) *list* of superblock types (names). The list items may be prefixed with "no" to specify the types which should be ignored. For example:

```
blkid --probe --match-types vfat,ext3,ext4 /dev/sda1
```

probes for vfat, ext3 and ext4 filesystems, and

```
blkid --probe --match-types nominix /dev/sda1
```

probes for all supported formats except minix filesystems. This option is only useful together with **--probe**.

-o, --output *format*

Use the specified output format. Note that the order of variables and devices is not fixed. See also option **-s**. The *format* parameter may be:

full

print all tags (the default)

value

print the value of the tags

list

print the devices in a user-friendly format; this output format is unsupported for low-level probing (**--probe** or **--info**).

This output format is **DEPRECATED** in favour of the **lsblk(8)** command.

device

print the device name only; this output format is always enabled for the **--label** and **--uuid** options

udev

print key="value" pairs for easy import into the udev environment; the keys are prefixed by ID_FS_ or ID_PART_ prefixes. The value may be modified to be safe for udev environment; allowed is plain ASCII, hex-escaping and valid UTF-8, everything else (including whitespaces) is replaced with '_'. The keys with _ENC postfix use hex-escaping for unsafe chars.

The udev output returns the ID_FS_AMBIVALENT tag if more superblocks are detected, and ID_PART_ENTRY_* tags are always returned for all partitions including empty partitions.

This output format is **DEPRECATED**.

export

print key=value pairs for easy import into the environment; this output format is automatically enabled when I/O Limits (**--info** option) are requested.

The non-printing characters are encoded by ^ and M- notation and all potentially unsafe characters are escaped.

-O, --offset *offset*

Probe at the given *offset* (only useful with **--probe**). This option can be used together with the **--info** option.

-p, --probe

Switch to low-level superblock probing mode (bypassing the cache).

Note that low-level probing also returns information about partition table type (PTTYPE tag) and partitions (PART_ENTRY_* tags). The tag names produced by low-level probing are based on names used internally by libblkid and it may be different than when executed without **--probe** (for example PART_ENTRY_UUID= vs PARTUUID=). See also **--no-part-details**.

-s, --match-tag *tag*

For each (specified) device, show only the tags that match *tag*. It is possible to specify multiple **--match-tag** options. If no tag is specified, then all tokens are shown for all (specified) devices. In order to just refresh the cache without showing any tokens, use **--match-tag none** with no other options.

-S, --size *size*

Override the size of device/file (only useful with **--probe**).

-t, --match-token *NAME=value*

Search for block devices with tokens named *NAME* that have the value *value*, and display any devices which are found. Common values for *NAME* include **TYPE**, **LABEL**, and **UUID**. If there are no devices specified on the command line, all block devices will be searched; otherwise only the specified devices are searched.

-u, --usages *list*

Restrict the probing functions to the specified (comma-separated) *list* of "usage" types. Supported usage types are: filesystem, raid, crypto and other. The list items may be prefixed with "no" to specify the usage types which should be ignored. For example:

blkid --probe --usages filesystem,other /dev/sda1

probes for all filesystem and other (e.g., swap) formats, and

blkid --probe --usages noraid /dev/sda1

probes for all supported formats except RAIDs. This option is only useful together with **--probe**.

-U, --uuid *uuid*

Look up the device that uses this filesystem *uuid*. For more details see the **--label** option.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

EXIT STATUS

If the specified device or device addressed by specified token (option **--match-token**) was found and it's possible to gather any information about the device, an exit status 0 is returned. Note the option **--match-tag** filters output tags, but it does not affect exit status.

If the specified token was not found, or no (specified) devices could be identified, or it is impossible to gather any information about the device identifiers or device content an exit status of 2 is returned.

For usage or other errors, an exit status of 4 is returned.

If an ambivalent probing result was detected by low-level probing mode (**-p**), an exit status of 8 is returned.

CONFIGURATION FILE

The standard location of the */etc/blkid.conf* config file can be overridden by the environment variable **BLKID_CONF**. The following options control the libblkid library:

SEND_UEVENT=<yes/not>

Sends uevent when */dev/disk/by-{label,uuid,partuuid,partlabel}*/ symlink does not match with LABEL, UUID, PARTUUID or PARTLABEL on the device. Default is "yes".

CACHE_FILE=<path>

Overrides the standard location of the cache file. This setting can be overridden by the environment variable **BLKID_FILE**. Default is */run/blkid/blkid.tab*, or */etc/blkid.tab* on systems without a */run* directory.

EVALUATE=<methods>

Defines LABEL and UUID evaluation method(s). Currently, the libblkid library supports the "udev" and "scan" methods. More than one method may be specified in a comma-separated list. Default is "udev,scan". The "udev" method uses udev */dev/disk/by-** symlinks and the "scan" method scans all block devices from the */proc/partitions* file.

ENVIRONMENT

Setting *LIBBLKID_DEBUG=all* enables debug output.

AUTHORS

blkid was written by Andreas Dilger for libblkid and improved by Theodore Ts'o and Karel Zak.

SEE ALSO

libblkid(3), **findfs(8)**, **lsblk(8)**, **wipefs(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **blkid** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

btrfs-man5 – topics about the BTRFS filesystem (mount options, supported file attributes and other)

DESCRIPTION

This document describes topics related to BTRFS that are not specific to the tools. Currently covers:

1. mount options
2. filesystem features
3. checksum algorithms
4. compression
5. filesystem exclusive operations
6. filesystem limits
7. bootloader support
8. file attributes
9. zoned mode
10. control device
11. filesystems with multiple block group profiles
12. seeding device
13. raid56 status and recommended practices
14. storage model

15.

hardware considerations

MOUNT OPTIONS

This section describes mount options specific to BTRFS. For the generic mount options please refer to **mount(8)** manpage. The options are sorted alphabetically (discarding the *no* prefix).

Note

most mount options apply to the whole filesystem and only options in the first mounted subvolume will take effect. This is due to lack of implementation and may change in the future. This means that (for example) you can't set per-subvolume *nodatacow*, *nodatasum*, or *compress* using mount options. This should eventually be fixed, but it has proved to be difficult to implement correctly within the Linux VFS framework.

Mount options are processed in order, only the last occurrence of an option takes effect and may disable other options due to constraints (see eg. *nodatacow* and *compress*). The output of *mount* command shows which options have been applied.

acl, noacl

(default: on)

Enable/disable support for Posix Access Control Lists (ACLs). See the **acl(5)** manual page for more information about ACLs.

The support for ACL is build-time configurable (BTRFS_FS_POSIX_ACL) and mount fails if *acl* is requested but the feature is not compiled in.

autodefrag, noautodefrag

(since: 3.0, default: off)

Enable automatic file defragmentation. When enabled, small random writes into files (in a range of tens of kilobytes, currently it's 64K) are detected and queued up for the defragmentation process. Not well suited for large database workloads.

The read latency may increase due to reading the adjacent blocks that make up the range for defragmentation, successive write will merge the blocks in the new location.

Warning

Defragmenting with Linux kernel versions < 3.9 or ≥ 3.14-rc2 as well as with Linux stable kernel versions ≥ 3.10.31, ≥ 3.12.12 or ≥ 3.13.4 will break up the reflinks of COW data (for example files copied with **cp --reflink**, snapshots or de-duplicated data). This may cause considerable increase of space usage depending on the broken up reflinks.

barrier, nobarrier

(default: on)

Ensure that all IO write operations make it through the device cache and are stored permanently when the filesystem is at its consistency checkpoint. This typically means that a flush command is sent to the device that will synchronize all pending data and ordinary metadata blocks, then writes the superblock and issues another flush.

The write flushes incur a slight hit and also prevent the IO block scheduler to reorder requests in a more effective way. Disabling barriers gets rid of that penalty but will most certainly lead to a corrupted filesystem in case of a crash or power loss. The ordinary metadata blocks could be yet unwritten at the time the new superblock is stored permanently, expecting that the block pointers to metadata were stored permanently before.

On a device with a volatile battery-backed write-back cache, the *nobarrier* option will not lead to filesystem corruption as the pending blocks are supposed to make it to the permanent storage.

check_int, check_int_data, check_int_print_mask=value

(since: 3.0, default: off)

These debugging options control the behavior of the integrity checking module (the BTRFS_FS_CHECK_INTEGRITY config option required). The main goal is to verify that all blocks from a given transaction period are properly linked.

check_int enables the integrity checker module, which examines all block write requests to ensure on-disk consistency, at a large memory and CPU cost.

check_int_data includes extent data in the integrity checks, and implies the *check_int* option.

check_int_print_mask takes a bitmask of BTRFSIC_PRINT_MASK_* values as defined in *fs/btrfs/check-integrity.c*, to control the integrity checker module behavior.

See comments at the top of *fs/btrfs/check-integrity.c* for more information.

clear_cache

Force clearing and rebuilding of the disk space cache if something has gone wrong. See also: *space_cache*.

commit=seconds

(since: 3.12, default: 30)

Set the interval of periodic transaction commit when data are synchronized to permanent storage. Higher interval values lead to larger amount of unwritten data, which has obvious consequences when the system crashes. The upper bound is not forced, but a warning is printed if it's more than 300 seconds (5 minutes). Use with care.

compress, compress=type[:level], compress-force, compress-force=type[:level]

(default: off, level support since: 5.1)

Control BTRFS file data compression. Type may be specified as *zlib*, *lzo*, *zstd* or *no* (for no compression, used for remounting). If no type is specified, *zlib* is used. If *compress-force* is specified, then compression will always be attempted, but the data may end up uncompressed if the compression would make them larger.

Both *zlib* and *zstd* (since version 5.1) expose the compression level as a tunable knob with higher levels trading speed and memory (*zstd*) for higher compression ratios. This can be set by appending a colon and the desired level. Zlib accepts the range [1, 9] and zstd accepts [1, 15]. If no level is set, both currently use a default level of 3. The value 0 is an alias for the default level.

Otherwise some simple heuristics are applied to detect an incompressible file. If the first blocks written to a file are not compressible, the whole file is permanently marked to skip compression. As this is too simple, the *compress-force* is a workaround that will compress most of the files at the cost of some wasted CPU cycles on failed attempts. Since kernel 4.15, a set of heuristic algorithms have been improved by using frequency sampling, repeated pattern detection and Shannon entropy calculation to avoid that.

Note

If compression is enabled, *nodatacow* and *nodatasum* are disabled.

datacow, nodatacow

(default: on)

Enable data copy-on-write for newly created files. *Nodacow* implies *nodatasum*, and disables *compression*. All files created under *nodacow* are also set the NOCOW file attribute (see **chattr(1)**).

Note

If *nodacow* or *nodatasum* are enabled, compression is disabled.

Updates in-place improve performance for workloads that do frequent overwrites, at the cost of potential partial writes, in case the write is interrupted (system crash, device failure).

datasum, nodatasum

(default: on)

Enable data checksumming for newly created files. *Datasum* implies *datacow*, ie. the normal mode of operation. All files created under *nodatasum* inherit the "no checksums" property, however there's no corresponding file attribute (see **chattr(1)**).

Note

If *nodacow* or *nodatasum* are enabled, compression is disabled.

There is a slight performance gain when checksums are turned off, the corresponding metadata blocks holding the checksums do not need to be updated. The cost of checksumming of the blocks in memory is much lower than the IO, modern CPUs feature hardware support of the checksumming algorithm.

degraded

(default: off)

Allow mounts with less devices than the RAID profile constraints require. A read-write mount (or remount) may fail when there are too many devices missing, for example if a stripe member is completely missing from RAID0.

Since 4.14, the constraint checks have been improved and are verified on the chunk level, not at the device level. This allows degraded mounts of filesystems with mixed RAID profiles for data and metadata, even if the device number constraints would not be satisfied for some of the profiles.

Example: metadata — raid1, data — single, devices — /dev/sda, /dev/sdb

Suppose the data are completely stored on *sda*, then missing *sdb* will not prevent the mount, even if 1 missing device would normally prevent (any) *single* profile to mount. In case some of the data chunks are stored on *sdb*, then the constraint of single/data is not satisfied and the filesystem cannot be mounted.

device=devicepath

Specify a path to a device that will be scanned for BTRFS filesystem during mount. This is usually done automatically by a device manager (like udev) or using the **btrfs device scan** command (eg. run from the initial ramdisk). In cases where this is not possible the *device* mount option can help.

Note

booting eg. a RAID1 system may fail even if all filesystem's *device* paths are provided as the actual device nodes may not be discovered by the system at that point.

discard, discard=sync, discard=async, nodiscard

(default: off, async support since: 5.6)

Enable discarding of freed file blocks. This is useful for SSD devices, thinly provisioned LUNs, or virtual machine images; however, every storage layer must support discard for it to work.

In the synchronous mode (*sync* or without option value), lack of asynchronous queued TRIM on the backing device TRIM can severely degrade performance, because a synchronous TRIM operation will be attempted instead. Queued TRIM requires newer than SATA revision 3.1 chipsets and devices.

The asynchronous mode (*async*) gathers extents in larger chunks before sending them to the devices for TRIM. The overhead and performance impact should be negligible compared to the previous mode and it's supposed to be the preferred mode if needed.

If it is not necessary to immediately discard freed blocks, then the **fstrim** tool can be used to discard all free blocks in a batch. Scheduling a TRIM during a period of low system activity will prevent latent interference with the performance of other operations. Also, a device may ignore the TRIM command if the range is too small, so running a batch discard has a greater probability of actually discarding the blocks.

enospc_debug, noenospc_debug

(default: off)

Enable verbose output for some ENOSPC conditions. It's safe to use but can be noisy if the system reaches near-full state.

fatal_errors=*action*

(since: 3.4, default: bug)

Action to take when encountering a fatal error.

bug

BUG() on a fatal error, the system will stay in the crashed state and may be still partially usable, but reboot is required for full operation

panic

panic() on a fatal error, depending on other system configuration, this may be followed by a reboot. Please refer to the documentation of kernel boot parameters, eg. *panic*, *oops* or *crashkernel*.

flushoncommit, noflushoncommit

(default: off)

This option forces any data dirtied by a write in a prior transaction to commit as part of the current commit, effectively a full filesystem sync.

This makes the committed state a fully consistent view of the file system from the application's perspective (i.e. it includes all completed file system operations). This was previously the behavior only when a snapshot was created.

When off, the filesystem is consistent but buffered writes may last more than one transaction commit.

fragment=type

(depends on compile-time option BTRFS_DEBUG, since: 4.4, default: off)

A debugging helper to intentionally fragment given *type* of block groups. The type can be *data*, *metadata* or *all*. This mount option should not be used outside of debugging environments and is not recognized if the kernel config option *BTRFS_DEBUG* is not enabled.

nologreplay

(default: off, even read-only)

The tree-log contains pending updates to the filesystem until the full commit. The log is replayed on next mount, this can be disabled by this option. See also *treelog*. Note that *nologreplay* is the same as *norecovery*.

Warning

currently, the tree log is replayed even with a read-only mount! To disable that behaviour, mount also with *nologreplay*.

max_inline=bytes

(default: min(2048, page size))

Specify the maximum amount of space, that can be inlined in a metadata B-tree leaf. The value is specified in bytes, optionally with a K suffix (case insensitive). In practice, this value is limited by the filesystem block size (named *sectorsize* at mkfs time), and memory page size of the system. In case of *sectorsize* limit, there's some space unavailable due to leaf headers. For example, a 4k *sectorsize*, maximum size of inline data is about 3900 bytes.

Inlining can be completely turned off by specifying 0. This will increase data block slack if file sizes are much smaller than block size but will reduce metadata consumption in return.

Note

the default value has changed to 2048 in kernel 4.6.

metadata_ratio=value

(default: 0, internal logic)

Specifies that 1 metadata chunk should be allocated after every *value* data chunks. Default behaviour depends on internal logic, some percent of unused metadata space is attempted to be maintained but is not always possible if there's not enough space left for chunk allocation. The option could be useful to override the internal logic in favor of the metadata allocation if the expected workload is supposed to be metadata intense (snapshots, reflinks, xattrs, inlined files).

norecovery

(since: 4.5, default: off)

Do not attempt any data recovery at mount time. This will disable *logreplay* and avoids other write operations. Note that this option is the same as *nologreplay*.

Note

The opposite option *recovery* used to have different meaning but was changed for consistency with other filesystems, where *norecovery* is used for skipping log replay. BTRFS does the same and in general will try

to avoid any write operations.

rescan_uuid_tree

(since: 3.12, default: off)

Force check and rebuild procedure of the UUID tree. This should not normally be needed.

rescue

(since: 5.9)

Modes allowing mount with damaged filesystem structures.

-
- *usebackuproot* (since: 5.9, replaces standalone option *usebackuproot*)
-
- *nologreplay* (since: 5.9, replaces standalone option *nologreplay*)
-
- *ignorebadroots, ibadroots* (since: 5.11)
-
- *ignoredatacsums, idatacsums* (since: 5.11)
-
- *all* (since: 5.9)

skip_balance

(since: 3.3, default: off)

Skip automatic resume of an interrupted balance operation. The operation can later be resumed with **btrfs balance resume**, or the paused state can be removed with **btrfs balance cancel**. The default behaviour is to resume an interrupted balance immediately after a volume is mounted.

space_cache, space_cache=version, nospace_cache

(*nospace_cache* since: 3.2, *space_cache=v1* and *space_cache=v2* since 4.5, default: *space_cache=v1*)

Options to control the free space cache. The free space cache greatly improves performance when reading block group free space into memory. However, managing the space cache consumes some resources, including a small amount of disk space.

There are two implementations of the free space cache. The original one, referred to as *v1*, is the safe default. The *v1* space cache can be disabled at mount time with *nospace_cache* without clearing.

On very large filesystems (many terabytes) and certain workloads, the performance of the *v1* space cache may degrade drastically. The *v2* implementation, which adds a new B-tree called the free space tree, addresses this issue. Once enabled, the *v2* space cache will always be used and cannot be disabled unless it is cleared. Use *clear_cache,space_cache=v1* or *clear_cache,nospace_cache* to do so. If *v2* is enabled, kernels without *v2* support will only be able to mount the filesystem in read-only mode.

The **btrfs-check(8)** and **mkfs.btrfs(8)** commands have full *v2* free space cache support since v4.19.

If a version is not explicitly specified, the default implementation will be chosen, which is *v1*.

ssd, ssd_spread, nossd, nossd_spread

(default: SSD autodetected)

Options to control SSD allocation schemes. By default, BTRFS will enable or disable SSD optimizations depending on status of a device with respect to rotational or non-rotational type. This is determined by the contents of */sys/block/DEV/queue/rotational*. If it is 0, the *ssd* option is turned on. The option *nossd* will disable the autodetection.

The optimizations make use of the absence of the seek penalty that's inherent for the rotational devices. The blocks can be typically written faster and are not offloaded to separate threads.

Note

Since 4.14, the block layout optimizations have been dropped. This used to help with first generations of SSD devices. Their FTL (flash translation layer) was not effective and the optimization was supposed to improve the wear by better aligning blocks. This is no longer true with modern SSD devices and the optimization had no real benefit. Furthermore it caused increased fragmentation. The layout tuning has been kept intact for the option *ssd_spread*.

The *ssd_spread* mount option attempts to allocate into bigger and aligned chunks of unused space, and may perform better on low-end SSDs. *ssd_spread* implies *ssd*, enabling all other SSD heuristics as well. The option *nossd* will disable all SSD options while *nossd_spread* only disables *ssd_spread*.

subvol=*path*

Mount subvolume from *path* rather than the toplevel subvolume. The *path* is always treated as relative to the toplevel subvolume. This mount option overrides the default subvolume set for the given filesystem.

subvolid=subvolid

Mount subvolume specified by a *subvolid* number rather than the toplevel subvolume. You can use **btrfs subvolume list** or **btrfs subvolume show** to see subvolume ID numbers. This mount option overrides the default subvolume set for the given filesystem.

Note

if both *subvolid* and *subvol* are specified, they must point at the same subvolume, otherwise the mount will fail.

thread_pool=number

(default: min(NRPCUS + 2, 8))

The number of worker threads to start. NRPCUS is number of on-line CPUs detected at the time of mount. Small number leads to less parallelism in processing data and metadata, higher numbers could lead to a performance hit due to increased locking contention, process scheduling, cache-line bouncing or costly data transfers between local CPU memories.

treelog, notreelog

(default: on)

Enable the tree logging used for *fsync* and *O_SYNC* writes. The tree log stores changes without the need of a full filesystem sync. The log operations are flushed at sync and transaction commit. If the system crashes between two such syncs, the pending tree log operations are replayed during mount.

Warning

currently, the tree log is replayed even with a read-only mount! To disable that behaviour, also mount with *nologreplay*.

The tree log could contain new files/directories, these would not exist on a mounted filesystem if the log is not replayed.

usebackuproot

(since: 4.6, default: off)

Enable autorecovery attempts if a bad tree root is found at mount time. Currently this scans a backup list of several previous tree roots and tries to use the first readable. This can be used with read-only

mounts as well.

Note

This option has replaced *recovery*.

user_subvol_rm_allowed

(default: off)

Allow subvolumes to be deleted by their respective owner. Otherwise, only the root user can do that.

Note

historically, any user could create a snapshot even if he was not owner of the source subvolume, the subvolume deletion has been restricted for that reason. The subvolume creation has been restricted but this mount option is still required. This is a usability issue. Since 4.18, the **rmdir(2)** syscall can delete an empty subvolume just like an ordinary directory. Whether this is possible can be detected at runtime, see *rmdir_subvol* feature in *FILESYSTEM FEATURES*.

DEPRECATED MOUNT OPTIONS

List of mount options that have been removed, kept for backward compatibility.

recovery

(since: 3.2, default: off, deprecated since: 4.5)

Note

this option has been replaced by *usebackuproot* and should not be used but will work on 4.5+ kernels.

inode_cache, noinode_cache

(removed in: 5.11, since: 3.0, default: off)

Note

the functionality has been removed in 5.11, any stale data created by previous use of the *inode_cache* option can be removed by **btrfs check --clear-ino-cache**.

NOTES ON GENERIC MOUNT OPTIONS

Some of the general mount options from **mount(8)** that affect BTRFS and are worth mentioning.

noatime

under read intensive work–loads, specifying *noatime* significantly improves performance because no new access time information needs to be written. Without this option, the default is *relatime*, which only reduces the number of inode atime updates in comparison to the traditional *strictatime*. The worst case for atime updates under *relatime* occurs when many files are read whose atime is older than 24 h

and which are freshly snapshotted. In that case the atime is updated *and* COW happens – for each file – in bulk. See also <https://lwn.net/Articles/499293/> – *Atime and btrfs: a bad combination?* (LWN, 2012-05-31).

Note that *noatime* may break applications that rely on atime uptimes like the venerable Mutt (unless you use maildir mailboxes).

FILESYSTEM FEATURES

The basic set of filesystem features gets extended over time. The backward compatibility is maintained and the features are optional, need to be explicitly asked for so accidental use will not create incompatibilities.

There are several classes and the respective tools to manage the features:

at mkfs time only

This is namely for core structures, like the b-tree nodesize or checksum algorithm, see **mkfs.btrfs(8)** for more details.

after mkfs, on an unmounted filesystem

Features that may optimize internal structures or add new structures to support new functionality, see **btrfstune(8)**. The command **btrfs inspect–internal dump–super device** will dump a superblock, you can map the value of *incompat_flags* to the features listed below

after mkfs, on a mounted filesystem

The features of a filesystem (with a given UUID) are listed in **/sys/fs/btrfs/UUID/features/**, one file per feature. The status is stored inside the file. The value *1* is for enabled and active, while *0* means the feature was enabled at mount time but turned off afterwards.

Whether a particular feature can be turned on a mounted filesystem can be found in the directory **/sys/fs/btrfs/features/**, one file per feature. The value *1* means the feature can be enabled.

List of features (see also **mkfs.btrfs(8)** section *FILESYSTEM FEATURES*):

big_metadata

(since: 3.4)

the filesystem uses *nodesize* for metadata blocks, this can be bigger than the page size

compress_lzo

(since: 2.6.38)

the *lzo* compression has been used on the filesystem, either as a mount option or via **btrfs filesystem defrag**.

compress_zstd

(since: 4.14)

the *zstd* compression has been used on the filesystem, either as a mount option or via **btrfs filesystem defrag**.

default_subvol

(since: 2.6.34)

the default subvolume has been set on the filesystem

extended_iref

(since: 3.7)

increased hardlink limit per file in a directory to 65536, older kernels supported a varying number of hardlinks depending on the sum of all file name sizes that can be stored into one metadata block

free_space_tree

(since: 4.5)

free space representation using a dedicated b-tree, successor of v1 space cache

metadata_uuid

(since: 5.0)

the main filesystem UUID is the metadata_uuid, which stores the new UUID only in the superblock while all metadata blocks still have the UUID set at mkfs time, see **btrfstune(8)** for more

mixed_backref

(since: 2.6.31)

the last major disk format change, improved backreferences, now default

mixed_groups

(since: 2.6.37)

mixed data and metadata block groups, ie. the data and metadata are not separated and occupy the same block groups, this mode is suitable for small volumes as there are no constraints how the remaining space should be used (compared to the split mode, where empty metadata space cannot be used for data and vice versa)

on the other hand, the final layout is quite unpredictable and possibly highly fragmented, which means worse performance

no_holes

(since: 3.14)

improved representation of file extents where holes are not explicitly stored as an extent, saves a few percent of metadata if sparse files are used

raid1c34

(since: 5.5)

extended RAID1 mode with copies on 3 or 4 devices respectively

raid56

(since: 3.9)

the filesystem contains or contained a raid56 profile of block groups

rmdir_subvol

(since: 4.18)

indicate that **rmdir(2)** syscall can delete an empty subvolume just like an ordinary directory. Note that this feature only depends on the kernel version.**skinny_metadata**

(since: 3.10)

reduced-size metadata for extent references, saves a few percent of metadata

send_stream_version

(since: 5.10)

number of the highest supported send stream version

supported_checksums

(since: 5.5)

list of checksum algorithms supported by the kernel module, the respective modules or built-in implementing the algorithms need to be present to mount the filesystem, see *CHECKSUM ALGORITHMS***supported_sectorsizes**

(since: 5.13)

list of values that are accepted as sector sizes (**mkfs.btrfs --sectorsize**) by the running kernel

supported_rescue_options

(since: 5.11)

list of values for the mount option *rescue* that are supported by the running kernel, see **btrfs(5)**

zoned

(since: 5.12)

zoned mode is allocation/write friendly to host-managed zoned devices, allocation space is partitioned into fixed-size zones that must be updated sequentially, see *ZONED MODE*

SWAPFILE SUPPORT

The swapfile is supported since kernel 5.0. Use **swapon(8)** to activate the swapfile. There are some limitations of the implementation in btrfs and linux swap subsystem:

- filesystem – must be only single device
- filesystem – must have only *single* data profile
- swapfile – the containing subvolume cannot be snapshotted
- swapfile – must be preallocated
- swapfile – must be nodatacow (ie. also nodatasum)
- swapfile – must not be compressed

The limitations come namely from the COW-based design and mapping layer of blocks that allows the advanced features like relocation and multi-device filesystems. However, the swap subsystem expects simpler mapping and no background changes of the file blocks once they've been attached to swap.

With active swapfiles, the following whole–filesystem operations will skip swapfile extents or may fail:

-
- balance – block groups with swapfile extents are skipped and reported, the rest will be processed normally
-
- resize grow – unaffected
-
- resize shrink – works as long as the extents are outside of the shrunk range
-
- device add – a new device does not interfere with existing swapfile and this operation will work, though no new swapfile can be activated afterwards
-
- device delete – if the device has been added as above, it can be also deleted
-
- device replace – ditto

When there are no active swapfiles and a whole–filesystem exclusive operation is running (ie. balance, device delete, shrink), the swapfiles cannot be temporarily activated. The operation must finish first.

To create and activate a swapfile run the following commands:

```
# truncate -s 0 swapfile
# chattr +C swapfile
# fallocate -l 2G swapfile
# chmod 0600 swapfile
# mkswap swapfile
# swapon swapfile
```

Please note that the UUID returned by the *mkswap* utility identifies the swap "filesystem" and because it's stored in a file, it's not generally visible and usable as an identifier unlike if it was on a block device.

The file will appear in */proc/swaps*:

```
# cat /proc/swaps
Filename      Type      Size      Used      Priority
/path/swapfile  file    2097152     0      -2
```

The swapfile can be created as one–time operation or, once properly created, activated on each boot by the *swapon -a* command (usually started by the service manager). Add the following entry to */etc/fstab*, assuming the filesystem that provides the */path* has been already mounted at this point. Additional mount options relevant for the swapfile can be set too (like priority, not the btrfs mount options).

```
/path/swapfile    none    swap    defaults    0 0
```

CHECKSUM ALGORITHMS

There are several checksum algorithms supported. The default and backward compatible is *crc32c*. Since kernel 5.5 there are three more with different characteristics and trade-offs regarding speed and strength. The following list may help you to decide which one to select.

CRC32C (32bit digest)

default, best backward compatibility, very fast, modern CPUs have instruction-level support, not collision-resistant but still good error detection capabilities

XXHASH (64bit digest)

can be used as CRC32C successor, very fast, optimized for modern CPUs utilizing instruction pipelining, good collision resistance and error detection

SHA256 (256bit digest)

a cryptographic-strength hash, relatively slow but with possible CPU instruction acceleration or specialized hardware cards, FIPS certified and in wide use

BLAKE2b (256bit digest)

a cryptographic-strength hash, relatively fast with possible CPU acceleration using SIMD extensions, not standardized but based on BLAKE which was a SHA3 finalist, in wide use, the algorithm used is BLAKE2b-256 that's optimized for 64bit platforms

The *digest size* affects overall size of data block checksums stored in the filesystem. The metadata blocks have a fixed area up to 256bits (32 bytes), so there's no increase. Each data block has a separate checksum stored, with additional overhead of the b-tree leaves.

Approximate relative performance of the algorithms, measured against CRC32C using reference software implementations on a 3.5GHz intel CPU:

```
[ cols="^,>,>,>",width="50%" ]
```

Digest	Cycles/4KiB	Ratio	Implementation
CRC32C	1700	1.00	CPU instruction
XXHASH	2500	1.44	reference impl.
SHA256	105000	61	reference impl.
SHA256	36000	21	libgcrypt/AVX2
SHA256	63000	37	libsodium/AVX2
BLAKE2b	22000	13	reference impl.
BLAKE2b	19000	11	libgcrypt/AVX2
BLAKE2b	19000	11	libsodium/AVX2

Many kernels are configured with SHA256 as built-in and not as a module. The accelerated versions are however provided by the modules and must be loaded explicitly (**modprobe sha256**) before mounting the filesystem to make use of them. You can check in `/sys/fs/btrfs/FSID/checksum` which one is used. If you see `sha256-generic`, then you may want to unmount and mount the filesystem again, changing that on a mounted filesystem is not possible. Check the file `/proc/crypto`, when the implementation is built-in, you'd find

```
name      : sha256
driver    : sha256-generic
module    : kernel
priority  : 100
...

```

while accelerated implementation is e.g.

```
name      : sha256
driver    : sha256-avx2
module    : sha256_ssse3
priority  : 170
...

```

COMPRESSION

Btrfs supports transparent file compression. There are three algorithms available: ZLIB, LZO and ZSTD (since v4.14). Basically, compression is on a file by file basis. You can have a single btrfs mount point that has some files that are uncompressed, some that are compressed with LZO, some with ZLIB, for instance (though you may not want it that way, it is supported).

To enable compression, mount the filesystem with options `compress` or `compress-force`. Please refer to section *MOUNT OPTIONS*. Once compression is enabled, all new writes will be subject to compression. Some files may not compress very well, and these are typically not recompressed but still written

uncompressed.

Each compression algorithm has different speed/ratio trade offs. The levels can be selected by a mount option and affect only the resulting size (ie. no compatibility issues).

Basic characteristics:

ZLIB slower, higher compression ratio

-
- levels: 1 to 9, mapped directly, default level is 3
-
- good backward compatibility

LZO faster compression and decompression than zlib, worse compression ratio, designed to be fast

-
- no levels
-

good backward compatibility

ZSTD compression comparable to zlib with higher compression/decompression speeds and different ratio

-
- levels: 1 to 15
-

since 4.14, levels since 5.1

The differences depend on the actual data set and cannot be expressed by a single number or recommendation. Higher levels consume more CPU time and may not bring a significant improvement, lower levels are close to real time.

The algorithms could be mixed in one file as they're stored per extent. The compression can be changed on a file by **btrfs filesystem defrag** command, using the *-c* option, or by **btrfs property set** using the *compression* property. Setting compression by *chattr +c* utility will set it to zlib.

INCOMPRESSIBLE DATA

Files with already compressed data or with data that won't compress well with the CPU and memory constraints of the kernel implementations are using a simple decision logic. If the first portion of data being compressed is not smaller than the original, the compression of the file is disabled — unless the filesystem is mounted with *compress-force*. In that case compression will always be attempted on the file only to be later discarded. This is not optimal and subject to optimizations and further development.

If a file is identified as incompressible, a flag is set (NOCOMPRESS) and it's sticky. On that file compression won't be performed unless forced. The flag can be also set by *chattr +m* (since e2fsprogs 1.46.2) or by properties with value *no* or *none*. Empty value will reset it to the default that's currently applicable on the mounted filesystem.

There are two ways to detect incompressible data:

- actual compression attempt – data are compressed, if the result is not smaller, it's discarded, so this depends on the algorithm and level
- pre-compression heuristics – a quick statistical evaluation on the data is performed and based on the result either compression is performed or skipped, the NOCOMPRESS bit is not set just by the heuristic, only if the compression algorithm does not make an improvement

PRE-COMPRESSION HEURISTICS

The heuristics aim to do a few quick statistical tests on the compressed data in order to avoid probably costly compression that would turn out to be inefficient. Compression algorithms could have internal detection of incompressible data too but this leads to more overhead as the compression is done in another thread and has to write the data anyway. The heuristic is read-only and can utilize cached memory.

The tests performed based on the following: data sampling, long repeated pattern detection, byte frequency, Shannon entropy.

COMPATIBILITY WITH OTHER FEATURES

Compression is done using the COW mechanism so it's incompatible with *nodatacow*. Direct IO works on compressed files but will fall back to buffered writes. Currently *nodatasum* and compression don't work together.

FILESYSTEM EXCLUSIVE OPERATIONS

There are several operations that affect the whole filesystem and cannot be run in parallel. Attempt to start one while another is running will fail.

Since kernel 5.10 the currently running operation can be obtained from `/sys/fs/UUID/exclusive_operation` with following values and operations:

- balance
- device add

- - device delete
- - device replace
- - resize
- - swapfile activate
- - none

Enqueuing is supported for several btrfs subcommands so they can be started at once and then serialized.

FILESYSTEM LIMITS

maximum file name length

255

maximum symlink target length

depends on the *nodesize* value, for 4k it's 3949 bytes, for larger nodesize it's 4095 due to the system limit PATH_MAX

The symlink target may not be a valid path, ie. the path name components can exceed the limits (NAME_MAX), there's no content validation at **symlink(3)** creation.

maximum number of inodes

264 but depends on the available metadata space as the inodes are created dynamically

inode numbers

minimum number: 256 (for subvolumes), regular files and directories: 257

maximum file length

inherent limit of btrfs is 264 (16 EiB) but the linux VFS limit is 263 (8 EiB)

maximum number of subvolumes

the subvolume ids can go up to 264 but the number of actual subvolumes depends on the available metadata space, the space consumed by all subvolume metadata includes bookkeeping of shared extents can be large (MiB, GiB)

maximum number of hardlinks of a file in a directory

65536 when the **extref** feature is turned on during mkfs (default), roughly 100 otherwise

minimum filesystem size

the minimal size of each device depends on the *mixed-bg* feature, without that (the default) it's about 109MiB, with mixed-bg it's 16MiB

BOOTLOADER SUPPORT

GRUB2 (<https://www.gnu.org/software/grub>) has the most advanced support of booting from BTRFS with respect to features.

U-boot (<https://www.denx.de/wiki/U-Boot/>) has decent support for booting but not all BTRFS features are implemented, check the documentation.

EXTLINUX (from the <https://syslinux.org> project) can boot but does not support all features. Please check the upstream documentation before you use it.

The first 1MiB on each device is unused with the exception of primary superblock that is on the offset 64KiB and spans 4KiB.

FILE ATTRIBUTES

The btrfs filesystem supports setting file attributes or flags. Note there are old and new interfaces, with confusing names. The following list should clarify that:

-

attributes: **chattr(1)** or **lsattr(1)** utilities (the ioctls are FS_IOC_GETFLAGS and FS_IOC_SETFLAGS), due to the ioctl names the attributes are also called flags

-

xflags: to distinguish from the previous, it's extended flags, with tunable bits similar to the attributes but extensible and new bits will be added in the future (the ioctls are FS_IOC_FSGETXATTR and FS_IOC_FSSETXATTR but they are not related to extended attributes that are also called xattrs), there's no standard tool to change the bits, there's support in **xfs_io(8)** as command **xfs_io -c chattr**

ATTRIBUTES**a**

append only, new writes are always written at the end of the file

A

no atime updates

c

compress data, all data written after this attribute is set will be compressed. Please note that compression is also affected by the mount options or the parent directory attributes.

When set on a directory, all newly created files will inherit this attribute. This attribute cannot be set with *m* at the same time.

C

no copy-on-write, file data modifications are done in-place

When set on a directory, all newly created files will inherit this attribute.

Note

due to implementation limitations, this flag can be set/unset only on empty files.

d

no dump, makes sense with 3rd party tools like **dump(8)**, on BTRFS the attribute can be set/unset but no other special handling is done

D

synchronous directory updates, for more details search **open(2)** for *O_SYNC* and *O_DSYNC*

i

immutable, no file data and metadata changes allowed even to the root user as long as this attribute is

set (obviously the exception is unsetting the attribute)

m

no compression, permanently turn off compression on the given file. Any compression mount options will not affect this file. (**chattr** support added in 1.46.2)

When set on a directory, all newly created files will inherit this attribute. This attribute cannot be set with *c* at the same time.

S

synchronous updates, for more details search **open(2)** for *O_SYNC* and *O_DSYNC*

No other attributes are supported. For the complete list please refer to the **chattr(1)** manual page.

XFLAGS

There's overlap of letters assigned to the bits with the attributes, this list refers to what **xfs_io(8)** provides:

i

immutable, same as the attribute

a

append only, same as the attribute

s

synchronous updates, same as the attribute *S*

A

no atime updates, same as the attribute

d

no dump, same as the attribute

ZONED MODE

Since version 5.12 btrfs supports so called *zoned mode*. This is a special on-disk format and allocation/write strategy that's friendly to zoned devices. In short, a device is partitioned into fixed-size zones and each zone can be updated by append-only manner, or reset. As btrfs has no fixed data structures, except the super blocks, the zoned mode only requires block placement that follows the device constraints. You can learn about the whole architecture at <https://zonedstorage.io>.

The devices are also called SMR/ZBC/ZNS, in *host-managed* mode. Note that there are devices that appear as non-zoned but actually are, this is *drive-managed* and using zoned mode won't help.

The zone size depends on the device, typical sizes are 256MiB or 1GiB. In general it must be a power of two. Emulated zoned devices like *null_blk* allow to set various zone sizes.

REQUIREMENTS, LIMITATIONS

- all devices must have the same zone size
- maximum zone size is 8GiB
- mixing zoned and non-zoned devices is possible, the zone writes are emulated, but this is namely for testing
- the super block is handled in a special way and is at different locations than on a non-zoned filesystem:
 - primary: 0B (and the next two zones)
 - secondary: 512G (and the next two zones)
 - tertiary: 4TiB (4096GiB, and the next two zones)

INCOMPATIBLE FEATURES

The main constraint of the zoned devices is lack of in-place update of the data. This is inherently incompatible with some features:

- nodatacow – overwrite in-place, cannot create such files
- fallocate – preallocating space for in-place first write

-
- mixed-bg – unordered writes to data and metadata, fixing that means using separate data and metadata block groups
-
- booting – the zone at offset 0 contains superblock, resetting the zone would destroy the bootloader data

Initial support lacks some features but they're planned:

-
- only single profile is supported
-
- fstrim – due to dependency on free space cache v1

SUPER BLOCK

As said above, super block is handled in a special way. In order to be crash safe, at least one zone in a known location must contain a valid superblock. This is implemented as a ring buffer in two consecutive zones, starting from known offsets 0, 512G and 4TiB. The values are different than on non-zoned devices. Each new super block is appended to the end of the zone, once it's filled, the zone is reset and writes continue to the next one. Looking up the latest super block needs to read offsets of both zones and determine the last written version.

The amount of space reserved for super block depends on the zone size. The secondary and tertiary copies are at distant offsets as the capacity of the devices is expected to be large, tens of terabytes. Maximum zone size supported is 8GiB, which would mean that eg. offset 0–16GiB would be reserved just for the super block on a hypothetical device of that zone size. This is wasteful but required to guarantee crash safety.

CONTROL DEVICE

There's a character special device **/dev/btrfs-control** with major and minor numbers 10 and 234 (the device can be found under the *misc* category).

```
$ ls -l /dev/btrfs-control
crw----- 1 root root 10, 234 Jan 1 12:00 /dev/btrfs-control
```

The device accepts some ioctl calls that can perform following actions on the filesystem module:

-
- scan devices for btrfs filesystem (ie. to let multi-device filesystems mount automatically) and register them with the kernel module
-
- similar to scan, but also wait until the device scanning process is finished for a given filesystem
-
- get the supported features (can be also found under */sys/fs/btrfs/features*)

The device is created when btrfs is initialized, either as a module or a built-in functionality and makes sense only in connection with that. Running eg. mkfs without the module loaded will not register the device and will probably warn about that.

In rare cases when the module is loaded but the device is not present (most likely accidentally deleted), it's possible to recreate it by

```
# mknod --mode=600 /dev/btrfs-control c 10 234
```

or (since 5.11) by a convenience command

```
# btrfs rescue create-control-device
```

The control device is not strictly required but the device scanning will not work and a workaround would need to be used to mount a multi-device filesystem. The mount option *device* can trigger the device scanning during mount, see also **btrfs device scan**.

FILESYSTEM WITH MULTIPLE PROFILES

It is possible that a btrfs filesystem contains multiple block group profiles of the same type. This could happen when a profile conversion using balance filters is interrupted (see **btrfs-balance(8)**). Some *btrfs* commands perform a test to detect this kind of condition and print a warning like this:

WARNING: Multiple block group profiles detected, see 'man btrfs(5)'.

WARNING: Data: single, raid1

WARNING: Metadata: single, raid1

The corresponding output of **btrfs filesystem df** might look like:

WARNING: Multiple block group profiles detected, see 'man btrfs(5)'.

WARNING: Data: single, raid1

WARNING: Metadata: single, raid1

Data, RAID1: total=832.00MiB, used=0.00B

Data, single: total=1.63GiB, used=0.00B

System, single: total=4.00MiB, used=16.00KiB

Metadata, single: total=8.00MiB, used=112.00KiB

Metadata, RAID1: total=64.00MiB, used=32.00KiB

GlobalReserve, single: total=16.25MiB, used=0.00B

There's more than one line for type *Data* and *Metadata*, while the profiles are *single* and *RAID1*.

This state of the filesystem OK but most likely needs the user/administrator to take an action and finish the interrupted tasks. This cannot be easily done automatically, also the user knows the expected final profiles.

In the example above, the filesystem started as a single device and *single* block group profile. Then another device was added, followed by balance with *convert=raid1* but for some reason hasn't finished. Restarting the balance with *convert=raid1* will continue and end up with filesystem with all block group profiles *RAID1*.

Note

If you're familiar with balance filters, you can use *convert=raid1,profiles=single,soft*, which will take only the unconverted *single* profiles and convert them to *raid1*. This may speed up the conversion as it would not try to

rewrite the already convert *raid1* profiles.

Having just one profile is desired as this also clearly defines the profile of newly allocated block groups, otherwise this depends on internal allocation policy. When there are multiple profiles present, the order of selection is RAID6, RAID5, RAID10, RAID1, RAID0 as long as the device number constraints are satisfied.

Commands that print the warning were chosen so they're brought to user attention when the filesystem state is being changed in that regard. This is: *device add*, *device delete*, *balance cancel*, *balance pause*. Commands that report space usage: *filesystem df*, *device usage*. The command *filesystem usage* provides a line in the overall summary:

Multiple profiles:	yes (data, metadata)
--------------------	----------------------

SEEDING DEVICE

The COW mechanism and multiple devices under one hood enable an interesting concept, called a seeding device: extending a read-only filesystem on a single device filesystem with another device that captures all writes. For example imagine an immutable golden image of an operating system enhanced with another device that allows to use the data from the golden image and normal operation. This idea originated on CD-ROMs with base OS and allowing to use them for live systems, but this became obsolete. There are technologies providing similar functionality, like *unionmount*, *overlays* or *qcow2* image snapshot.

The seeding device starts as a normal filesystem, once the contents is ready, **btrfs tune -S 1** is used to flag it as a seeding device. Mounting such device will not allow any writes, except adding a new device by **btrfs device add**. Then the filesystem can be remounted as read-write.

Given that the filesystem on the seeding device is always recognized as read-only, it can be used to seed multiple filesystems, at the same time. The UUID that is normally attached to a device is automatically changed to a random UUID on each mount.

Once the seeding device is mounted, it needs the writable device. After adding it, something like *remount -o remount,rw /path* makes the filesystem at */path* ready for use. The simplest usecase is to throw away all changes by unmounting the filesystem when convenient.

Alternatively, deleting the seeding device from the filesystem can turn it into a normal filesystem, provided that the writable device can also contain all the data from the seeding device.

The seeding device flag can be cleared again by **btrfs tune -f -s 0**, eg. allowing to update with newer data but please note that this will invalidate all existing filesystems that use this particular seeding device. This works for some usecases, not for others, and a forcing flag to the command is mandatory to avoid accidental mistakes.

Example how to create and use one seeding device:

```
# mkfs.btrfs /dev/sda
# mount /dev/sda /mnt/mnt1
# ... fill mnt1 with data
# umount /mnt/mnt1
# btrfs tune -S 1 /dev/sda
# mount /dev/sda /mnt/mnt1
# btrfs device add /dev/sdb /mnt
# mount -o remount,rw /mnt/mnt1
# ... /mnt/mnt1 is now writable
```

Now `/mnt/mnt1` can be used normally. The device `/dev/sda` can be mounted again with another writable device:

```
# mount /dev/sda /mnt/mnt2
# btrfs device add /dev/sdc /mnt/mnt2
# mount -o remount,rw /mnt/mnt2
# ... /mnt/mnt2 is now writable
```

The writable device (`/dev/sdb`) can be decoupled from the seeding device and used independently:

```
# btrfs device delete /dev/sda /mnt/mnt1
```

As the contents originated in the seeding device, it's possible to turn `/dev/sdb` to a seeding device again and repeat the whole process.

A few things to note:

- it's recommended to use only single device for the seeding device, it works for multiple devices but the *single* profile must be used in order to make the seeding device deletion work
- block group profiles *single* and *dup* support the usecases above
- the label is copied from the seeding device and can be changed by **btrfs filesystem label**
- each new mount of the seeding device gets a new random UUID

RAID56 STATUS AND RECOMMENDED PRACTICES

The RAID56 feature provides striping and parity over several devices, same as the traditional RAID5/6. There are some implementation and design deficiencies that make it unreliable for some corner cases and the feature **should not be used in production, only for evaluation or testing**. The power failure safety for metadata with RAID56 is not 100%.

Metadata

Do not use `raid5` nor `raid6` for metadata. Use `raid1` or `raid1c3` respectively.

The substitute profiles provide the same guarantees against loss of 1 or 2 devices, and in some respect can be an improvement. Recovering from one missing device will only need to access the remaining 1st or 2nd copy, that in general may be stored on some other devices due to the way RAID1 works on btrfs, unlike on a striped profile (similar to `raid0`) that would need all devices all the time.

The space allocation pattern and consumption is different (eg. on N devices): for `raid5` as an example, a 1GiB chunk is reserved on each device, while with `raid1` there's each 1GiB chunk stored on 2 devices. The consumption of each 1GiB of used metadata is then $N * 1GiB$ for vs $2 * 1GiB$. Using `raid1` is also more convenient for balancing/converting to other profile due to lower requirement on the available chunk space.

Missing/incomplete support

When RAID56 is on the same filesystem with different raid profiles, the space reporting is inaccurate, eg. `df`, `btrfs filesystem df` or `btrfs filesystem usage`. When there's only one profile per block group type (eg. raid5 for data) the reporting is accurate.

When scrub is started on a RAID56 filesystem, it's started on all devices that degrade the performance. The workaround is to start it on each device separately. Due to that the device stats may not match the actual state and some errors might get reported multiple times.

The *write hole* problem.

STORAGE MODEL

A storage model is a model that captures key physical aspects of data structure in a data store. A filesystem is the logical structure organizing data on top of the storage device.

The filesystem assumes several features or limitations of the storage device and utilizes them or applies measures to guarantee reliability. BTRFS in particular is based on a COW (copy on write) mode of writing, ie. not updating data in place but rather writing a new copy to a different location and then atomically switching the pointers.

In an ideal world, the device does what it promises. The filesystem assumes that this may not be true so additional mechanisms are applied to either detect misbehaving hardware or get valid data by other means. The devices may (and do) apply their own detection and repair mechanisms but we won't assume any.

The following assumptions about storage devices are considered (sorted by importance, numbers are for further reference):

1.

atomicity of reads and writes of blocks/sectors (the smallest unit of data the device presents to the upper layers)

2.

there's a flush command that instructs the device to forcibly order writes before and after the command; alternatively there's a barrier command that facilitates the ordering but may not flush the data

3.

data sent to write to a given device offset will be written without further changes to the data and to the offset

4.

writes can be reordered by the device, unless explicitly serialized by the flush command

5.

reads and writes can be freely reordered and interleaved

The consistency model of BTRFS builds on these assumptions. The logical data updates are grouped, into a generation, written on the device, serialized by the flush command and then the super block is written ending the generation. All logical links among metadata comprising a consistent view of the data may not cross the generation boundary.

WHEN THINGS GO WRONG

No or partial atomicity of block reads/writes (1)

-
- Problem:* a partial block contents is written (*torn write*), eg. due to a power glitch or other electronics failure during the read/write
-
- Detection:* checksum mismatch on read
-
- Repair:* use another copy or rebuild from multiple blocks using some encoding scheme

The flush command does not flush (2)

This is perhaps the most serious problem and impossible to mitigate by filesystem without limitations and design restrictions. What could happen in the worst case is that writes from one generation bleed to another one, while still letting the filesystem consider the generations isolated. Crash at any point would leave data on the device in an inconsistent state without any hint what exactly got written, what is missing and leading to stale metadata link information.

Devices usually honor the flush command, but for performance reasons may do internal caching, where the flushed data are not yet persistently stored. A power failure could lead to a similar scenario as above, although it's less likely that later writes would be written before the cached ones. This is beyond what a filesystem can take into account. Devices or controllers are usually equipped with batteries or capacitors to write the cache contents even after power is cut. (*Battery backed write cache*)

Data get silently changed on write (3)

Such thing should not happen frequently, but still can happen spuriously due the complex internal workings of devices or physical effects of the storage media itself.

-
- Problem:* while the data are written atomically, the contents get changed
-
- Detection:* checksum mismatch on read
-
- Repair:* use another copy or rebuild from multiple blocks using some encoding scheme

Data get silently written to another offset (3)

This would be another serious problem as the filesystem has no information when it happens. For that reason the measures have to be done ahead of time. This problem is also commonly called *ghost write*.

The metadata blocks have the checksum embedded in the blocks, so a correct atomic write would not corrupt the checksum. It's likely that after reading such block the data inside would not be consistent with the rest. To rule that out there's embedded block number in the metadata block. It's the logical block number because this is what the logical structure expects and verifies.

HARDWARE CONSIDERATIONS

The following is based on information publicly available, user feedback, community discussions or bug report analyses. It's not complete and further research is encouraged when in doubt.

MAIN MEMORY

The data structures and raw data blocks are temporarily stored in computer memory before they get written to the device. It is critical that memory is reliable because even simple bit flips can have vast consequences and lead to damaged structures, not only in the filesystem but in the whole operating system.

Based on experience in the community, memory bit flips are more common than one would think. When it happens, it's reported by the tree-checker or by a checksum mismatch after reading blocks. There are some very obvious instances of bit flips that happen, e.g. in an ordered sequence of keys in metadata blocks. We can easily infer from the other data what values get damaged and how. However, fixing that is not straightforward and would require cross-referencing data from the entire filesystem to see the scope.

If available, ECC memory should lower the chances of bit flips, but this type of memory is not available in all cases. A memory test should be performed in case there's a visible bit flip pattern, though this may not detect a faulty memory module because the actual load of the system could be the factor making the problems appear. In recent years attacks on how the memory modules operate have been demonstrated (*rowhammer*) achieving specific bits to be flipped. While these were targeted, this shows that a series of reads or writes can affect unrelated parts of memory.

Further reading:

-

https://en.wikipedia.org/wiki/Row_hammer

What to do:

-

run *memtest*, note that sometimes memory errors happen only when the system is under heavy load that the default memtest cannot trigger

-

memory errors may appear as filesystem going read-only due to "pre write" check, that verify meta data before they get written but fail some basic consistency checks

DIRECT MEMORY ACCESS (DMA)

Another class of errors is related to DMA (direct memory access) performed by device drivers. While this could be considered a software error, the data transfers that happen without CPU assistance may accidentally corrupt other pages. Storage devices utilize DMA for performance reasons, the filesystem structures and data pages are passed back and forth, making errors possible in case page life time is not properly tracked.

There are lots of quirks (device-specific workarounds) in Linux kernel drivers (regarding not only DMA) that are added when found. The quirks may avoid specific errors or disable some features to avoid worse problems.

What to do:

-

- use up-to-date kernel (recent releases or maintained long term support versions)

-

- as this may be caused by faulty drivers, keep the systems up-to-date

ROTATIONAL DISKS (HDD)

Rotational HDDs typically fail at the level of individual sectors or small clusters. Read failures are caught on the levels below the filesystem and are returned to the user as *EIO – Input/output error*. Reading the blocks repeatedly may return the data eventually, but this is better done by specialized tools and filesystem takes the result of the lower layers. Rewriting the sectors may trigger internal remapping but this inevitably leads to data loss.

Disk firmware is technically software but from the filesystem perspective is part of the hardware. IO requests are processed, and caching or various other optimizations are performed, which may lead to bugs under high load or unexpected physical conditions or unsupported use cases.

Disks are connected by cables with two ends, both of which can cause problems when not attached properly. Data transfers are protected by checksums and the lower layers try hard to transfer the data correctly or not at all. The errors from badly-connecting cables may manifest as large amount of failed read or write requests, or as short error bursts depending on physical conditions.

What to do:

-

- check *smartctl* for potential issues

SOLID STATE DRIVES (SSD)

The mechanism of information storage is different from HDDs and this affects the failure mode as well. The data are stored in cells grouped in large blocks with limited number of resets and other write constraints. The firmware tries to avoid unnecessary resets and performs optimizations to maximize the storage media lifetime. The known techniques are deduplication (blocks with same fingerprint/hash are mapped to same physical block), compression or internal remapping and garbage collection of used memory cells. Due to the additional processing there are measures to verify the data e.g. by ECC codes.

The observations of failing SSDs show that the whole electronic fails at once or affects a lot of data (e.g. stored on one chip). Recovering such data may need specialized equipment and reading data repeatedly does not help as it's possible with HDDs.

There are several technologies of the memory cells with different characteristics and price. The lifetime is directly affected by the type and frequency of data written. Writing "too much" distinct data (e.g. encrypted) may render the internal deduplication ineffective and lead to a lot of rewrites and increased wear of the memory cells.

There are several technologies and manufacturers so it's hard to describe them but there are some that exhibit similar behaviour:

-

- expensive SSD will use more durable memory cells and is optimized for reliability and high load

-

- cheap SSD is projected for a lower load ("desktop user") and is optimized for cost, it may employ the optimizations and/or extended error reporting partially or not at all

It's not possible to reliably determine the expected lifetime of an SSD due to lack of information about how it works or due to lack of reliable stats provided by the device.

Metadata writes tend to be the biggest component of lifetime writes to a SSD, so there is some value in reducing them. Depending on the device class (high end/low end) the features like DUP block group profiles may affect the reliability in both ways:

-
- *high end* are typically more reliable and using *single* for data and metadata could be suitable to reduce device wear
-
- *low end* could lack ability to identify errors so an additional redundancy at the filesystem level (checksums, *DUP*) could help

Only users who consume 50 to 100% of the SSD's actual lifetime writes need to be concerned by the write amplification of btrfs DUP metadata. Most users will be far below 50% of the actual lifetime, or will write the drive to death and discover how many writes 100% of the actual lifetime was. SSD firmware often adds its own write multipliers that can be arbitrary and unpredictable and dependent on application behavior, and these will typically have far greater effect on SSD lifespan than DUP metadata. It's more or less impossible to predict when a SSD will run out of lifetime writes to within a factor of two, so it's hard to justify wear reduction as a benefit.

Further reading:

-
- <https://www.snia.org/educational-library/ssd-and-deduplication-end-spinning-disk-2012>
-
- <https://www.snia.org/educational-library/realities-solid-state-storage-2013-2013>
-
- <https://www.snia.org/educational-library/ssd-performance-primer-2013>
-
- [https://www.snia.org/educational-library/how-controllers-maximize\(ssd-life\)-2013](https://www.snia.org/educational-library/how-controllers-maximize(ssd-life)-2013)

What to do:

-
- run *smartctl* or self-tests to look for potential issues
-
- keep the firmware up-to-date

NVM EXPRESS, NON-VOLATILE MEMORY (NVMe)

NVMe is a type of persistent memory usually connected over a system bus (PCIe) or similar interface and the speeds are an order of magnitude faster than SSD. It is also a non-rotating type of storage, and is not typically connected by a cable. It's not a SCSI type device either but rather a complete specification for

logical device interface.

In a way the errors could be compared to a combination of SSD class and regular memory. Errors may exhibit as random bit flips or IO failures. There are tools to access the internal log (*nvme log* and *nvme-cl*) for a more detailed analysis.

There are separate error detection and correction steps performed e.g. on the bus level and in most cases never making it to the filesystem level. Once this happens it could mean there's some systematic error like overheating or bad physical connection of the device. You may want to run self-tests (using *smartctl*).

- https://en.wikipedia.org/wiki/NVM_Express

- https://www.smartmontools.org/wiki/NVMe_Support

DRIVE FIRMWARE

Firmware is technically still software but embedded into the hardware. As all software has bugs, so does firmware. Storage devices can update the firmware and fix known bugs. In some cases it's possible to avoid certain bugs by quirks (device-specific workarounds) in Linux kernel.

A faulty firmware can cause wide range of corruptions from small and localized to large affecting lots of data. Self-repair capabilities may not be sufficient.

What to do:

- check for firmware updates in case there are known problems, note that updating firmware can be risky on itself

- use up-to-date kernel (recent releases or maintained long term support versions)

SD FLASH CARDS

There are a lot of devices with low power consumption and thus using storage media based on low power consumption too, typically flash memory stored on a chip enclosed in a detachable card package. An improperly inserted card may be damaged by electrical spikes when the device is turned on or off. The chips storing data in turn may be damaged permanently. All types of flash memory have a limited number of rewrites, so the data are internally translated by FTL (flash translation layer). This is implemented in firmware (technically a software) and prone to bugs that manifest as hardware errors.

Adding redundancy like using DUP profiles for both data and metadata can help in some cases but a full backup might be the best option once problems appear and replacing the card could be required as well.

HARDWARE AS THE MAIN SOURCE OF FILESYSTEM CORRUPTIONS

If you use unreliable hardware and don't know about that, don't blame the filesystem when it tells you.

SEE ALSO

acl(5), btrfs(8), chattr(1), fstrim(8), ioctl(2), mkfs.btrfs(8), mount(8), swapon(8)

NAME

btrfs – a toolbox to manage btrfs filesystems

SYNOPSIS

btrfs <command> [<args>]

DESCRIPTION

The **btrfs** utility is a toolbox for managing btrfs filesystems. There are command groups to work with subvolumes, devices, for whole filesystem or other specific actions. See section **COMMANDS**.

There are also standalone tools for some tasks like **btrfs-convert** or **btrfstune** that were separate historically and/or haven't been merged to the main utility. See section **STANDALONE TOOLS** for more details.

For other topics (mount options, etc) please refer to the separate manual page **btrfs(5)**.

COMMAND SYNTAX

Any command name can be shortened so long as the shortened form is unambiguous, however, it is recommended to use full command names in scripts. All command groups have their manual page named **btrfs-<group>**.

For example: it is possible to run **btrfs sub snaps** instead of **btrfs subvolume snapshot**. But **btrfs file s** is not allowed, because **file s** may be interpreted both as **filesystem show** and as **filesystem sync**.

If the command name is ambiguous, the list of conflicting options is printed.

For an overview of a given command use *btrfs command --help* or *btrfs [command...] --help --full* to print all available options.

COMMANDS

balance

Balance btrfs filesystem chunks across single or several devices.

See **btrfs–balance(8)** for details.

check

Do off-line check on a btrfs filesystem.

See **btrfs–check(8)** for details.

device

Manage devices managed by btrfs, including add/delete/scan and so on.

See **btrfs–device(8)** for details.

filesystem

Manage a btrfs filesystem, including label setting-sync and so on.

See **btrfs–filesystem(8)** for details.

inspect–internal

Debug tools for developers/hackers.

See **btrfs–inspect–internal(8)** for details.

property

Get/set a property from/to a btrfs object.

See **btrfs–property(8)** for details.

qgroup

Manage quota group(qgroup) for btrfs filesystem.

See **btrfs-qgroup(8)** for details.

quota

Manage quota on btrfs filesystem like enabling/rescan and etc.

See **btrfs-quota(8)** and **btrfs-qgroup(8)** for details.

receive

Receive subvolume data from stdin/file for restore and etc.

See **btrfs-receive(8)** for details.

replace

Replace btrfs devices.

See **btrfs-replace(8)** for details.

rescue

Try to rescue damaged btrfs filesystem.

See **btrfs-rescue(8)** for details.

restore

Try to restore files from a damaged btrfs filesystem.

See **btrfs-restore(8)** for details.

scrub

Scrub a btrfs filesystem.

See **btrfs-scrub(8)** for details.

send

Send subvolume data to stdout/file for backup and etc.

See **btrfs-send(8)** for details.

subvolume

Create/delete/list/manage btrfs subvolume.

See **btrfs-subvolume(8)** for details.

STANDALONE TOOLS

New functionality could be provided using a standalone tool. If the functionality proves to be useful, then the standalone tool is declared obsolete and its functionality is copied to the main tool. Obsolete tools are removed after a long (years) depreciation period.

Tools that are still in active use without an equivalent in **btrfs**:

btrfs-convert

in-place conversion from ext2/3/4 filesystems to btrfs

btrfstune

tweak some filesystem properties on a unmounted filesystem

btrfs-select-super

rescue tool to overwrite primary superblock from a spare copy

btrfs-find-root

rescue helper to find tree roots in a filesystem

Deprecated and obsolete tools:

btrfs-debug-tree

moved to **btrfs inspect-internal dump-tree**. Removed from source distribution.

btrfs-show-super

moved to **btrfs inspect-internal dump-super**, standalone removed.

btrfs-zero-log

moved to **btrfs rescue zero-log**, standalone removed.

For space-constrained environments, it's possible to build a single binary with functionality of several standalone tools. This is following the concept of busybox where the file name selects the functionality. This works for symlinks or hardlinks. The full list can be obtained by **btrfs help --box**.

EXIT STATUS

btrfs returns a zero exit status if it succeeds. Non zero is returned in case of failure.

AVAILABILITY

btrfs is part of btrfs-progs. Please refer to the btrfs wiki <http://btrfs.wiki.kernel.org> for further details.

SEE ALSO

btrfs(5), **btrfs-balance(8)**, **btrfs-check(8)**, **btrfs-convert(8)**, **btrfs-device(8)**, **btrfs-filesystem(8)**,
btrfs-inspect-internal(8), **btrfs-property(8)**, **btrfs-qgroup(8)**, **btrfs-quota(8)**, **btrfs-receive(8)**,
btrfs-replace(8), **btrfs-rescue(8)**, **btrfs-restore(8)**, **btrfs-scrub(8)**, **btrfs-send(8)**, **btrfs-subvolume(8)**,
btrfstune(8), **mkfs.btrfs(8)**

NAME

btrfs-check – check or repair a btrfs filesystem

SYNOPSIS

btrfs check [options] <device>

DESCRIPTION

The filesystem checker is used to verify structural integrity of a filesystem and attempt to repair it if requested. It is recommended to unmount the filesystem prior to running the check, but it is possible to start checking a mounted filesystem (see *--force*).

By default, **btrfs check** will not modify the device but you can reaffirm that by the option *--readonly*.

btrfsck is an alias of **btrfs check** command and is now deprecated.

Warning

Do not use *--repair* unless you are advised to do so by a developer or an experienced user, and then only after having accepted that no *fsck* successfully repair all types of filesystem corruption. Eg. some other software or hardware bugs can fatally damage a volume.

The structural integrity check verifies if internal filesystem objects or data structures satisfy the constraints, point to the right objects or are correctly connected together.

There are several cross checks that can detect wrong reference counts of shared extents, backreferences, missing extents of inodes, directory and inode connectivity etc.

The amount of memory required can be high, depending on the size of the filesystem, similarly the run time. Check the modes that can also affect that.

SAFE OR ADVISORY OPTIONS

-b|—backup

use the first valid set of backup roots stored in the superblock

This can be combined with *--super* if some of the superblocks are damaged.

--check-data-csum

verify checksums of data blocks

This expects that the filesystem is otherwise OK, and is basically an offline *scrub* that does not repair data from spare copies.

--chunk-root <bytenr>

use the given offset *bytenr* for the chunk tree root

-E|—subvol-extents <subvolid>

show extent state for the given subvolume

-p|—progress

indicate progress at various checking phases

-Q|—qgroup-report

verify qgroup accounting and compare against filesystem accounting

-r|—tree-root <bytenr>

use the given offset *bytenr* for the tree root

--readonly

(default) run in read-only mode, this option exists to calm potential panic when users are going to run the checker

-s|—super <superblock>

use 'superblock'th superblock copy, valid values are 0, 1 or 2 if the respective superblock offset is

within the device size

This can be used to use a different starting point if some of the primary superblock is damaged.

--clear-space-cache v1|v2

completely wipe all free space cache of given type

For free space cache *v1*, the *clear_cache* kernel mount option only rebuilds the free space cache for block groups that are modified while the filesystem is mounted with that option. Thus, using this option with *v1* makes it possible to actually clear the entire free space cache.

For free space cache *v2*, the *clear_cache* kernel mount option destroys the entire free space cache. This option, with *v2* provides an alternative method of clearing the free space cache that doesn't require mounting the filesystem.

--clear-ino-cache

remove leftover items pertaining to the deprecated inode map feature

DANGEROUS OPTIONS

--repair

enable the repair mode and attempt to fix problems where possible

Note

there's a warning and 10 second delay when this option is run without *--force* to give users a chance to think twice before running repair, the warnings in documentation have shown to be insufficient

--init-csum-tree

create a new checksum tree and recalculate checksums in all files

Note

Do not blindly use this option to fix checksum mismatch problems.

--init-extent-tree

build the extent tree from scratch

Note

Do not use unless you know what you're doing.

--mode <MODE>

select mode of operation regarding memory and IO

The *MODE* can be one of:

original

The metadata are read into memory and verified, thus the requirements are high on large filesystems and can even lead to out-of-memory conditions. The possible workaround is to export the block device over network to a machine with enough memory.

lowmem

This mode is supposed to address the high memory consumption at the cost of increased IO when it needs to re-read blocks. This may increase run time.

Note

lowmem mode does not work with *--repair* yet, and is still considered experimental.

--force

allow work on a mounted filesystem. Note that this should work fine on a quiescent or read-only mounted filesystem but may crash if the device is changed externally, eg. by the kernel module. Repair without mount checks is not supported right now.

This option also skips the delay and warning in the repair mode (see *--repair*).

EXIT STATUS

btrfs check returns a zero exit status if it succeeds. Non zero is returned in case of failure.

AVAILABILITY

btrfs is part of btrfs-progs. Please refer to the btrfs wiki <http://btrfs.wiki.kernel.org> for further details.

SEE ALSO

mkfs.btrfs(8), **btrfs-scrub(8)**, **btrfs-rescue(8)**

NAME

bzip2, **bunzip2** – a block-sorting file compressor, v1.0.8
bzcat – decompresses files to stdout
bzip2recover – recovers data from damaged bzip2 files

SYNOPSIS

```
bzip2 [ -cdfkqstvzVL123456789 ] [ filenames ... ]
bzip2 [ -h|--help ]
bunzip2 [ -fkvsVL ] [ filenames ... ]
bunzip2 [ -h|--help ]
bzcat [ -s ] [ filenames ... ]
bzcat [ -h|--help ]
bzip2recover filename
```

DESCRIPTION

bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78-based compressors, and approaches the performance of the PPM family of statistical compressors.

The command-line options are deliberately very similar to those of *GNU gzip*, but they are not identical.

bzip2 expects a list of file names to accompany the command-line flags. Each file is replaced by a compressed version of itself, with the name "original_name.bz2". Each compressed file has the same modification date, permissions, and, when possible, ownership as the corresponding original, so that these properties can be correctly restored at decompression time. File name handling is naive in the sense that there is no mechanism for preserving original file names, permissions, ownerships or dates in filesystems which lack these concepts, or have serious file name length restrictions, such as MS-DOS.

bzip2 and *bunzip2* will by default not overwrite existing files. If you want this to happen, specify the *-f* flag.

If no file names are specified, *bzip2* compresses from standard input to standard output. In this case, *bzip2* will decline to write compressed output to a terminal, as this would be entirely incomprehensible and therefore pointless.

bunzip2 (or *bzip2 -d*) decompresses all specified files. Files which were not created by *bzip2* will be detected and ignored, and a warning issued. *bzip2* attempts to guess the filename for the decompressed file from that of the compressed file as follows:

filename.bz2	becomes	filename
filename.bz	becomes	filename
filename.tbz2	becomes	filename.tar
filename.tbz	becomes	filename.tar
anyothername	becomes	anyothername.out

If the file does not end in one of the recognised endings, *.bz2*, *.bz*, *.tbz2* or *.tbz*, *bzip2* complains that it cannot guess the name of the original file, and uses the original name with *.out* appended.

As with compression, supplying no filenames causes decompression from standard input to standard output.

bunzip2 will correctly decompress a file which is the concatenation of two or more compressed files. The result is the concatenation of the corresponding uncompressed files. Integrity testing (*-t*) of concatenated compressed files is also supported.

You can also compress or decompress files to the standard output by giving the `-c` flag. Multiple files may be compressed and decompressed like this. The resulting outputs are fed sequentially to `stdout`. Compression of multiple files in this manner generates a stream containing multiple compressed file representations. Such a stream can be decompressed correctly only by *bzip2* version 0.9.0 or later. Earlier versions of *bzip2* will stop after decompressing the first file in the stream.

bzcat (or *bzip2 -dc*) decompresses all specified files to the standard output.

bzip2 will read arguments from the environment variables *BZIP2* and *BZIP*, in that order, and will process them before any arguments read from the command line. This gives a convenient way to supply default arguments.

Compression is always performed, even if the compressed file is slightly larger than the original. Files of less than about one hundred bytes tend to get larger, since the compression mechanism has a constant overhead in the region of 50 bytes. Random data (including the output of most file compressors) is coded at about 8.05 bits per byte, giving an expansion of around 0.5%.

As a self-check for your protection, *bzip2* uses 32-bit CRCs to make sure that the decompressed version of a file is identical to the original. This guards against corruption of the compressed data, and against undetected bugs in *bzip2* (hopefully very unlikely). The chances of data corruption going undetected is microscopic, about one chance in four billion for each file processed. Be aware, though, that the check occurs upon decompression, so it can only tell you that something is wrong. It can't help you recover the original uncompressed data. You can use *bzip2recover* to try to recover data from damaged files.

Return values: 0 for a normal exit, 1 for environmental problems (file not found, invalid flags, I/O errors, &c), 2 to indicate a corrupt compressed file, 3 for an internal consistency error (eg, bug) which caused *bzip2* to panic.

OPTIONS

-c --stdout

Compress or decompress to standard output.

-d --decompress

Force decompression. *bzip2*, *bunzip2* and *bzcat* are really the same program, and the decision about what actions to take is done on the basis of which name is used. This flag overrides that mechanism, and forces *bzip2* to decompress.

-z --compress

The complement to `-d`: forces compression, regardless of the invocation name.

-t --test

Check integrity of the specified file(s), but don't decompress them. This really performs a trial decompression and throws away the result.

-f --force

Force overwrite of output files. Normally, *bzip2* will not overwrite existing output files. Also forces *bzip2* to break hard links to files, which it otherwise wouldn't do.

bzip2 normally declines to decompress files which don't have the correct magic header bytes. If forced (`-f`), however, it will pass such files through unmodified. This is how GNU *gzip* behaves.

-k --keep

Keep (don't delete) input files during compression or decompression.

-s --small

Reduce memory usage, for compression, decompression and testing. Files are decompressed and tested using a modified algorithm which only requires 2.5 bytes per block byte. This means any file can be decompressed in 2300 k of memory, albeit at about half the normal speed.

During compression, `-s` selects a block size of 200 k, which limits memory use to around the same figure, at the expense of your compression ratio. In short, if your machine is low on memory (8 megabytes or less), use `-s` for everything. See MEMORY MANAGEMENT below.

`-q --quiet`

Suppress non-essential warning messages. Messages pertaining to I/O errors and other critical events will not be suppressed.

`-v --verbose`

Verbose mode -- show the compression ratio for each file processed. Further `-v`'s increase the verbosity level, spewing out lots of information which is primarily of interest for diagnostic purposes.

`-h --help`

Print a help message and exit.

`-L --license -V --version`

Display the software version, license terms and conditions.

`-1 (or --fast) to -9 (or --best)`

Set the block size to 100 k, 200 k ... 900 k when compressing. Has no effect when decompressing. See MEMORY MANAGEMENT below. The `--fast` and `--best` aliases are primarily for GNU gzip compatibility. In particular, `--fast` doesn't make things significantly faster. And `--best` merely selects the default behaviour.

`--` Treats all subsequent arguments as file names, even if they start with a dash. This is so you can handle files with names beginning with a dash, for example: `bzip2 -- myfilename`.

`--repetitive-fast --repetitive-best`

These flags are redundant in versions 0.9.5 and above. They provided some coarse control over the behaviour of the sorting algorithm in earlier versions, which was sometimes useful. 0.9.5 and above have an improved algorithm which renders these flags irrelevant.

MEMORY MANAGEMENT

bzip2 compresses large files in blocks. The block size affects both the compression ratio achieved, and the amount of memory needed for compression and decompression. The flags `-1` through `-9` specify the block size to be 100,000 bytes through 900,000 bytes (the default) respectively. At decompression time, the block size used for compression is read from the header of the compressed file, and *bunzip2* then allocates itself just enough memory to decompress the file. Since block sizes are stored in compressed files, it follows that the flags `-1` to `-9` are irrelevant to and so ignored during decompression.

Compression and decompression requirements, in bytes, can be estimated as:

Compression: $400 \text{ k} + (8 \times \text{block size})$

Decompression: $100 \text{ k} + (4 \times \text{block size})$, or
 $100 \text{ k} + (2.5 \times \text{block size})$

Larger block sizes give rapidly diminishing marginal returns. Most of the compression comes from the first two or three hundred k of block size, a fact worth bearing in mind when using *bzip2* on small machines. It is also important to appreciate that the decompression memory requirement is set at compression time by the choice of block size.

For files compressed with the default 900 k block size, *bunzip2* will require about 3700 kbytes to decompress. To support decompression of any file on a 4 megabyte machine, *bunzip2* has an option to decompress using approximately half this amount of memory, about 2300 kbytes. Decompression speed is also halved, so you should use this option only where necessary. The relevant flag is `-s`.

In general, try and use the largest block size memory constraints allow, since that maximises the

compression achieved. Compression and decompression speed are virtually unaffected by block size.

Another significant point applies to files which fit in a single block -- that means most files you'd encounter using a large block size. The amount of real memory touched is proportional to the size of the file, since the file is smaller than a block. For example, compressing a file 20,000 bytes long with the flag -9 will cause the compressor to allocate around 7600 k of memory, but only touch 400 k + 20000 * 8 = 560 kbytes of it. Similarly, the decompressor will allocate 3700 k but only touch 100 k + 20000 * 4 = 180 kbytes.

Here is a table which summarises the maximum memory usage for different block sizes. Also recorded is the total compressed size for 14 files of the Calgary Text Compression Corpus totalling 3,141,622 bytes. This column gives some feel for how compression varies with block size. These figures tend to underestimate the advantage of larger block sizes for larger files, since the Corpus is dominated by smaller files.

Flag	Compress usage	Decompress usage	Decompress -s usage	Corpus Size
-1	1200k	500k	350k	914704
-2	2000k	900k	600k	877703
-3	2800k	1300k	850k	860338
-4	3600k	1700k	1100k	846899
-5	4400k	2100k	1350k	845160
-6	5200k	2500k	1600k	838626
-7	6100k	2900k	1850k	834096
-8	6800k	3300k	2100k	828642
-9	7600k	3700k	2350k	828642

RECOVERING DATA FROM DAMAGED FILES

bzip2 compresses files in blocks, usually 900 kbytes long. Each block is handled independently. If a media or transmission error causes a multi-block .bz2 file to become damaged, it may be possible to recover data from the undamaged blocks in the file.

The compressed representation of each block is delimited by a 48-bit pattern, which makes it possible to find the block boundaries with reasonable certainty. Each block also carries its own 32-bit CRC, so damaged blocks can be distinguished from undamaged ones.

bzip2recover is a simple program whose purpose is to search for blocks in .bz2 files, and write each block out into its own .bz2 file. You can then use *bzip2* -t to test the integrity of the resulting files, and decompress those which are undamaged.

bzip2recover takes a single argument, the name of the damaged file, and writes a number of files "rec00001file.bz2", "rec00002file.bz2", etc., containing the extracted blocks. The output filenames are designed so that the use of wildcards in subsequent processing -- for example, "bzip2 -dc rec*file.bz2 > recovered_data" -- processes the files in the correct order.

bzip2recover should be of most use dealing with large .bz2 files, as these will contain many blocks. It is clearly futile to use it on damaged single-block files, since a damaged block cannot be recovered. If you wish to minimise any potential data loss through media or transmission errors, you might consider compressing with a smaller block size.

PERFORMANCE NOTES

The sorting phase of compression gathers together similar strings in the file. Because of this, files containing very long runs of repeated symbols, like "aabbaabaab ..." (repeated several hundred times) may compress more slowly than normal. Versions 0.9.5 and above fare much better than previous versions in this respect. The ratio between worst-case and average-case compression time is in the region of 10:1. For

previous versions, this figure was more like 100:1. You can use the `-vvvv` option to monitor progress in great detail, if you want.

Decompression speed is unaffected by these phenomena.

bzip2 usually allocates several megabytes of memory to operate in, and then charges all over it in a fairly random fashion. This means that performance, both for compressing and decompressing, is largely determined by the speed at which your machine can service cache misses. Because of this, small changes to the code to reduce the miss rate have been observed to give disproportionately large performance improvements. I imagine *bzip2* will perform best on machines with very large caches.

CAVEATS

I/O error messages are not as helpful as they could be. *bzip2* tries hard to detect I/O errors and exit cleanly, but the details of what the problem is sometimes seem rather misleading.

This manual page pertains to version 1.0.8 of *bzip2*. Compressed data created by this version is entirely forwards and backwards compatible with the previous public releases, versions 0.1pl2, 0.9.0, 0.9.5, 1.0.0, 1.0.1, 1.0.2 and above, but with the following exception: 0.9.0 and above can correctly decompress multiple concatenated compressed files. 0.1pl2 cannot do this; it will stop after decompressing just the first file in the stream.

bzip2recover versions prior to 1.0.2 used 32-bit integers to represent bit positions in compressed files, so they could not handle compressed files more than 512 megabytes long. Versions 1.0.2 and above use 64-bit ints on some platforms which support them (GNU supported targets, and Windows). To establish whether or not *bzip2recover* was built with such a limitation, run it without arguments. In any event you can build yourself an unlimited version if you can recompile it with `MaybeUInt64` set to be an unsigned 64-bit integer.

AUTHOR

Julian Seward, jseward@acm.org.

<https://sourceware.org/bzip2/>

The ideas embodied in *bzip2* are due to (at least) the following people: Michael Burrows and David Wheeler (for the block sorting transformation), David Wheeler (again, for the Huffman coder), Peter Fenwick (for the structured coding model in the original *bzip*, and many refinements), and Alistair Moffat, Radford Neal and Ian Witten (for the arithmetic coder in the original *bzip*). I am much indebted for their help, support and advice. See the manual in the source distribution for pointers to sources of documentation. Christian von Roques encouraged me to look for faster sorting algorithms, so as to speed up compression. Bela Lubkin encouraged me to improve the worst-case compression performance. Donna Robinson XMLised the documentation. The bz* scripts are derived from those of GNU gzip. Many people sent patches, helped with portability problems, lent machines, gave advice and were generally helpful.

NAME

cal, ncal — displays a calendar and the date of Easter

SYNOPSIS

```
cal [-3hjy] [-A number] [-B number] [[month] year]
cal [-3hj] [-A number] [-B number] -m month [year]
ncal [-3bhjJpwySM] [-A number] [-B number] [-W number] [-s country_code]
      [[month] year]
ncal [-Jeo] [-A number] [-B number] [year]
ncal [-CN] [-H YYYY-mm-dd] [-d YYYY-mm]
```

DESCRIPTION

The **cal** utility displays a simple calendar in traditional format and **ncal** offers an alternative layout, more options and the date of Easter. The new format is a little cramped but it makes a year fit on a 25x80 terminal. If arguments are not specified, the current month is displayed.

The options are as follows:

- h** Turns off highlighting of today.
- J** Display Julian Calendar, if combined with the **-o** option, display date of Orthodox Easter according to the Julian Calendar.
- e** Display date of Easter (for western churches).
- j** Display Julian days (days one-based, numbered from January 1).
- m month** Display the specified *month*. If *month* is specified as a decimal number, appending ‘f’ or ‘p’ displays the same month of the following or previous year respectively.
- o** Display date of Orthodox Easter (Greek and Russian Orthodox Churches).
- p** Print the country codes and switching days from Julian to Gregorian Calendar as they are assumed by **ncal**. The country code as determined from the local environment is marked with an asterisk.
- s country_code** Assume the switch from Julian to Gregorian Calendar at the date associated with the *country_code*. If not specified, **ncal** tries to guess the switch date from the local environment or falls back to September 2, 1752. This was when Great Britain and her colonies switched to the Gregorian Calendar.
- w** Print the number of the week below each week column.
- y** Display a calendar for the specified year. This option is implied when a year but no month are specified on the command line.
- 3** Display the previous, current and next month surrounding today.
- 1** Display only the current month. This is the default.
- A number** Months to add after. The specified number of months is added to the end of the display. This is in addition to any date range selected by the **-y**, **-3**, or **-1** options. For example, “**cal -y -B2 -A2**” shows everything from November of the previous year to February of the following year. Negative numbers are allowed, in which case the specified number of months is subtracted. For example, “**cal -y -B-6**” shows July to December. And “**cal -A11**” simply shows the next 12 months.

-B *number*

Months to add before. The specified number of months is added to the beginning of the display. See **-A** for examples.

-C

Completely switch to **cal** mode. For **cal** like output only, use **-b** instead.

-d *YYYY-mm*

Use *YYYY-mm* as the current date (for debugging of date selection).

-H *YYYY-mm-dd*

Use *YYYY-mm-dd* as the current date (for debugging of highlighting).

-M

Weeks start on Monday.

-S

Weeks start on Sunday.

-W *number*

First week of the year has at least *number* days.

-b

Use oldstyle format for ncal output.

A single parameter specifies the year (1–9999) to be displayed; note the year must be fully specified: “**cal 89**” will not display a calendar for 1989. Two parameters denote the month and year; the month is either a number between 1 and 12, or a full or abbreviated name as specified by the current locale. Month and year default to those of the current system clock and time zone (so “**cal -m 8**” will display a calendar for the month of August in the current year).

Not all options can be used together. For example the options **-y**, **-3**, and **-1** are mutually exclusive. If inconsistent options are given, the later ones take precedence over the earlier ones.

A year starts on January 1.

Highlighting of dates is disabled if stdout is not a tty.

SEE ALSO

[calendar\(3\)](#), [strftime\(3\)](#)

STANDARDS

The **cal** utility is compliant with the X/Open System Interfaces option of the IEEE Std 1003.1-2008 (“POSIX.1”) specification.

The flags [**-3hyJeopw**], as well as the ability to specify a month name as a single argument, are extensions to that specification.

The week number computed by **-w** is compliant with the ISO 8601 specification.

HISTORY

A **cal** command appeared in Version 1 AT&T UNIX. Then **ncal** command appeared in FreeBSD 2.2.6. The output of the **cal** command is supposed to be bit for bit compatible to the original Unix **cal** command, because its output is processed by other programs like CGI scripts, that should not be broken. Therefore it will always output 8 lines, even if only 7 contain data. This extra blank line also appears with the original **cal** command, at least on Solaris 8

AUTHORS

The **ncal** command and manual were written by Wolfgang Helbig <helbig@FreeBSD.org>.

BUGS

The assignment of Julian–Gregorian switching dates to country codes is historically naive for many countries.

Not all options are compatible and using them in different orders will give varying results.

It is not possible to display Monday as the first day of the week with **cal**.

NAME

cat – concatenate files and print on the standard output

SYNOPSIS

cat [*OPTION*]... [*FILE*]...

DESCRIPTION

Concatenate FILE(s) to standard output.

With no FILE, or when FILE is **-**, read standard input.

-A, --show-all

equivalent to **-vET**

-b, --number-nonblank

number nonempty output lines, overrides **-n**

-e equivalent to **-vE**

-E, --show-ends

display \$ at end of each line

-n, --number

number all output lines

-s, --squeeze-blank

suppress repeated empty output lines

-t equivalent to **-vT**

-T, --show-tabs

display TAB characters as ^I

-u (ignored)

-v, --show-nonprinting

use ^ and M- notation, except for LFD and TAB

--help display this help and exit

--version

output version information and exit

EXAMPLES

cat f - g

Output f's contents, then standard input, then g's contents.

cat Copy standard input to standard output.

AUTHOR

Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

tac(1)

Full documentation <<https://www.gnu.org/software/coreutils/cat>>
or available locally via: info '(coreutils) cat invocation'

NAME

cfdisk – display or manipulate a disk partition table

SYNOPSIS

cfdisk [options] [*device*]

DESCRIPTION

cfdisk is a curses-based program for partitioning any block device. The default device is */dev/sda*.

Note that **cfdisk** provides basic partitioning functionality with a user-friendly interface. If you need advanced features, use **fdisk(8)** instead.

All disk label changes will remain in memory only, and the disk will be unmodified until you decide to write your changes. Be careful before using the write command.

Since version 2.25 **cfdisk** supports MBR (DOS), GPT, SUN and SGI disk labels, but no longer provides any functionality for CHS (Cylinder–Head–Sector) addressing. CHS has never been important for Linux, and this addressing concept does not make any sense for new devices.

Since version 2.25 **cfdisk** also does not provide a 'print' command any more. This functionality is provided by the utilities **partx(8)** and **lsblk(8)** in a very comfortable and rich way.

If you want to remove an old partition table from a device, use **wipefs(8)**.

OPTIONS

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

-L, --color[=when]

Colorize the output. The optional argument *when* can be **auto**, **never** or **always**. If the *when* argument is omitted, it defaults to **auto**. The colors can be disabled, for the current built-in default see **--help** output. See also the **COLORS** section.

--lock[=mode]

Use exclusive BSD lock for device or file it operates. The optional argument *mode* can be **yes**, **no** (or 1 and 0) or **nonblock**. If the *mode* argument is omitted, it defaults to **yes**. This option overwrites environment variable **\$LOCK_BLOCK_DEVICE**. The default is not to use any lock at all, but it's recommended to avoid collisions with **systemd-udevd(8)** or other tools.

-r, --read-only

Forced open in read-only mode.

-z, --zero

Start with an in-memory zeroed partition table. This option does not zero the partition table on the disk; rather, it simply starts the program without reading the existing partition table. This option allows you to create a new partition table from scratch or from an **sfdisk(8)**-compatible script.

COMMANDS

The commands for **cfdisk** can be entered by pressing the corresponding key (pressing *Enter* after the command is not necessary). Here is a list of the available commands:

b

Toggle the bootable flag of the current partition. This allows you to select which primary partition is bootable on the drive. This command may not be available for all partition label types.

d

Delete the current partition. This will convert the current partition into free space and merge it with any free space immediately surrounding the current partition. A partition already marked as free space or marked as unusable cannot be deleted.

h

Show the help screen.

n

Create a new partition from free space. **cfdisk** then prompts you for the size of the partition you want to create. The default size is equal to the entire available free space at the current position.

The size may be followed by a multiplicative suffix: KiB (=1024), MiB (=1024*1024), and so on for GiB, TiB, PiB, EiB, ZiB and YiB (the "iB" is optional, e.g., "K" has the same meaning as "KiB").

q

Quit the program. This will exit the program without writing any data to the disk.

r

Reduce or enlarge the current partition. **cfdisk** then prompts you for the new size of the partition. The default size is the current size. A partition marked as free space or marked as unusable cannot be resized.

Note that reducing the size of a partition might destroy data on that partition.

s

Sort the partitions in ascending start-sector order. When deleting and adding partitions, it is likely that the numbering of the partitions will no longer match their order on the disk. This command restores that match.

t

Change the partition type. By default, new partitions are created as *Linux* partitions.

u

Dump the current in-memory partition table to an **sfdisk(8)**-compatible script file.

The script files are compatible between **cfdisk**, **fdisk(8)** **sfdisk(8)** and other libfdisk applications. For more details see **sfdisk(8)**.

It is also possible to load an **sfdisk**-script into **cfdisk** if there is no partition table on the device or when you start **cfdisk** with the **--zero** command-line option.

W

Write the partition table to disk (you must enter an uppercase W). Since this might destroy data on the disk, you must either confirm or deny the write by entering 'yes' or 'no'. If you enter 'yes', **cfdisk** will write the partition table to disk and then tell the kernel to re-read the partition table from the disk.

The re-reading of the partition table does not always work. In such a case you need to inform the kernel about any new partitions by using **partprobe(8)** or **partx(8)**, or by rebooting the system.

x

Toggle extra information about a partition.

Up Arrow, Down Arrow

Move the cursor to the previous or next partition. If there are more partitions than can be displayed on a screen, you can display the next (previous) set of partitions by moving down (up) at the last (first) partition displayed on the screen.

Left Arrow, Right Arrow

Select the preceding or the next menu item. Hitting *Enter* will execute the currently selected item.

All commands can be entered with either uppercase or lowercase letters (except for **Write**). When in a submenu or at a prompt, you can hit the *Esc* key to return to the main menu.

COLORS

The output colorization is implemented by **terminal-colors.d(5)** functionality. Implicit coloring can be disabled by an empty file

/etc/terminal-colors.d/cfdisk.disable

for the **cfdisk** command or for all tools by

/etc/terminal-colors.d/disable

The user-specific *\$XDG_CONFIG_HOME/terminal-colors.d* or *\$HOME/.config/terminal-colors.d* overrides the global setting.

Note that the output colorization may be enabled by default, and in this case *terminal-colors.d* directories do not have to exist yet.

cfdisk does not support color customization with a color-scheme file.

ENVIRONMENT

CFDISK_DEBUG=all

enables cfdisk debug output.

LIBFDISK_DEBUG=all

enables libfdisk debug output.

LIBBLKID_DEBUG=all

enables libblkid debug output.

LIBSMARTCOLS_DEBUG=all

enables libsmartcols debug output.

LIBSMARTCOLS_DEBUG_PADDING=on

use visible padding characters. Requires enabled **LIBSMARTCOLS_DEBUG**.

LOCK_BLOCK_DEVICE=<mode>

use exclusive BSD lock. The mode is "1" or "0". See **--lock** for more details.

AUTHORS

Karel Zak <kzak@redhat.com>

The current **cfdisk** implementation is based on the original **cfdisk** from Kevin E. Martin <martin@cs.unc.edu>.

SEE ALSO

fdisk(8), parted(8), partprobe(8), partx(8), sfdisk(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **cfdisk** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

cgdisk – Curses-based GUID partition table (GPT) manipulator

SYNOPSIS

cgdisk [-a] *device*

DESCRIPTION

GPT fdisk is a text-mode family of programs for creation and manipulation of partition tables. The **cgdisk** member of this family employs a curses-based user interface for interaction using a text-mode menuing system. It will automatically convert an old-style Master Boot Record (MBR) partition table or BSD disklabel stored without an MBR carrier partition to the newer Globally Unique Identifier (GUID) Partition Table (GPT) format, or will load a GUID partition table. Other members of this program family are **gdisk** (the most feature-rich program of the group, with a non-curses-based interactive user interface) and **sgdisk** (which is driven via command-line options for use by experts or in scripts). FixParts is a related program for fixing a limited set of problems with MBR disks.

For information on MBR vs. GPT, as well as GPT terminology and structure, see the extended GPT fdisk documentation at <http://www.rodsbooks.com/gdisk/> or consult Wikipedia.

The **cgdisk** program employs a user interface similar to that of Linux's **cfdisk**, but **cgdisk** modifies GPT partitions. It also has the capability of transforming MBR partitions or BSD disklabels into GPT partitions. Like the original **cfdisk** program, **cgdisk** does not modify disk structures until you explicitly write them to disk, so if you make a mistake, you can exit from the program with the Quit option to leave your partitions unmodified.

Ordinarily, **cgdisk** operates on disk device files, such as */dev/sda* or */dev/hda* under Linux, */dev/disk0* under Mac OS X, or */dev/ad0* or */dev/da0* under FreeBSD. The program can also operate on disk image files, which can be either copies of whole disks (made with **dd**, for instance) or raw disk images used by emulators such as QEMU or VMWare. Note that only *raw* disk images are supported; **cgdisk** cannot work on compressed or other advanced disk image formats.

Upon start, **cgdisk** attempts to identify the partition type in use on the disk. If it finds valid GPT data, **cgdisk** will use it. If **cgdisk** finds a valid MBR or BSD disklabel but no GPT data, it will attempt to convert the MBR or disklabel into GPT form. (BSD disklabels are likely to have unusable first and/or final partitions because they overlap with the GPT data structures, though.) Upon exiting with the 'w' option, **cgdisk** replaces the MBR or disklabel with a GPT. *This action is potentially dangerous!* Your system may become unbootable, and partition type codes may become corrupted if the disk uses unrecognized type codes. Boot problems are particularly likely if you're multi-booting with any GPT-unaware OS. If you mistakenly launch **cgdisk** on an MBR disk, you can safely exit the program without making any changes by using the Quit option.

When creating a fresh partition table, certain considerations may be in order:

- * For data (non-boot) disks, and for boot disks used on BIOS-based computers with GRUB as the boot loader, partitions may be created in whatever order and in whatever sizes are desired.
- * Boot disks for EFI-based systems require an *EFI System Partition* (GPT fdisk internal code 0xEF00) formatted as FAT-32. The recommended size of this partition is between 100 and 300 MiB. Boot-related files are stored here. (Note that GNU Parted identifies such partitions as having the "boot flag" set.)
- * The GRUB 2 boot loader for BIOS-based systems makes use of a *BIOS Boot Partition* (GPT fdisk internal code 0xEF02), in which the secondary boot loader is stored, without the benefit of a

filesystem. This partition can typically be quite small (roughly 32 KiB to 1 MiB), but you should consult your boot loader documentation for details.

- * If Windows is to boot from a GPT disk, a partition of type *Microsoft Reserved* (GPT fdisk internal code 0x0C01) is recommended. This partition should be about 128 MiB in size. It ordinarily follows the EFI System Partition and immediately precedes the Windows data partitions. (Note that old versions of GNU Parted create all FAT partitions as this type, which actually makes the partition unusable for normal file storage in both Windows and Mac OS X.)
- * Some OSes' GPT utilities create some blank space (typically 128 MiB) after each partition. The intent is to enable future disk utilities to use this space. Such free space is not required of GPT disks, but creating it may help in future disk maintenance. You can use GPT fdisk's relative partition positioning option (specifying the starting sector as '+128M', for instance) to simplify creating such gaps.

OPTIONS

Only one command-line option is accepted, aside from the device filename: *-a*. This option alters the highlighting of partitions and blocks of free space: Instead of using ncurses, when *-a* is used **cgdisk** uses a ">" symbol to the left of the selected partition or free space. This option is intended for use on limited display devices such as teletypes and screen readers.

Interactions with **cgdisk** occur with its interactive text-mode menus. The display is broken into two interactive parts:

- * The partition display area, in which partitions and gaps between them (marked as "free space") are summarized.
- * The option selection area, in which buttons for the main options appear.

In addition, the top of the display shows the program's name and version number, the device filename associated with the disk, and the disk's size in both sectors and IEEE-1541 units (GiB, TiB, and so on).

You can use the following keys to move among the various options and to select among them:

up arrow

This key moves the partition selection up by one partition.

down arrow

This key moves the partition selection down by one partition.

Page Up

This key moves the partition selection up by one screen.

Page Down

This key moves the partition selection down by one screen.

right arrow

This key moves the option selection to the right by one item.

left arrow

This key moves the option selection to the left by one item.

- Enter** This key activates the currently selected option. You can also activate an option by typing the capitalized letter in the option's name on the keyboard, such as **a** to activate the Align option.

If more partitions exist than can be displayed in one screen, you can scroll between screens using the partition selection keys, much as in a text editor.

Available options are as described below. (Note that **cgdisk** provides a much more limited set of options than its sibling **gdisk**. If you need to perform partition table recovery, hybrid MBR modification, or other advanced operations, you should consult the **gdisk** documentation.)

- Align** Change the sector alignment value. Disks with more logical sectors than physical sectors (such as modern Advanced Format drives), some RAID configurations, and many SSD devices, can suffer performance problems if partitions are not aligned properly for their internal data structures. On new disks, GPT fdisk attempts to align partitions on 1 MiB boundaries (2048–sectors on disks with 512-byte sectors) by default, which optimizes performance for all of these disk types. On pre-partitioned disks, GPT fdisk attempts to identify the alignment value used on that disk, but will set 8-sector alignment on disks larger than 300 GB even if lesser alignment values are detected. In either case, it can be changed by using this option.

Backup

Save partition data to a backup file. You can back up your current in–memory partition table to a disk file using this option. The resulting file is a binary file consisting of the protective MBR, the main GPT header, the backup GPT header, and one copy of the partition table, in that order. Note that the backup is of the current in–memory data structures, so if you launch the program, make changes, and then use this option, the backup will reflect your changes.

- Delete** Delete a partition. This action deletes the entry from the partition table but does not disturb the data within the sectors originally allocated to the partition on the disk. If a corresponding hybrid MBR partition exists, **gdisk** deletes it, as well, and expands any adjacent 0xEE (EFI GPT) MBR protective partition to fill the new free space.

- Help** Print brief descriptions of all the options.

- Info** Show detailed partition information. The summary information shown in the partition display area necessarily omits many details, such as the partitions' unique GUIDs and the partitions' sector-exact start and end points. The Info option displays this information for a single partition.

- Load** Load partition data from a backup file. This option is the reverse of the Backup option. Note that restoring partition data from anything but the original disk is not recommended.

- naMe** Change the GPT name of a partition. This name is encoded as a UTF–16 string, but proper entry and display of anything beyond basic ASCII values requires suitable locale and font support. For the most part, Linux ignores the partition name, but it may be important in some OSes. GPT fdisk sets a default name based on the partition type code. Note that the GPT partition name is different from the filesystem name, which is encoded in the filesystem's data structures. Note also that to activate this item by typing its alphabetic equivalent, you must use **M**, not the more obvious **N**, because the latter is used by the next option....

- New** Create a new partition. You enter a starting sector, a size, a type code, and a name. The start sector can be specified in absolute terms as a sector number or as a position measured in kibibytes (K), mebibytes (M), gibibytes (G), tebibytes (T), or pebibytes (P); for instance, **40M** specifies a position 40MiB from the start of the disk. You can specify locations relative to the start or end of the specified default range by preceding the number by a '+' symbol, as in **+2G** to specify a point 2GiB after the default start sector. The size value can use the K, M, G, T, and P suffixes, too. Pressing the Enter key with no input specifies the default value, which is the start of the largest available block for the start sector and the full available size for the size.
- Quit** Quit from the program *without saving your changes*. Use this option if you just wanted to view information or if you make a mistake and want to back out of all your changes.
- Type** Change a single partition's type code. You enter the type code using a two-byte hexadecimal number. You may also enter a GUID directly, if you have one and **cgdisk** doesn't know it. If you don't know the type code for your partition, you can type **L** to see a list of known type codes. The type code list may optionally be filtered by a search string; for instance, entering **linux** shows only partition type codes with descriptions that include the string *Linux*. This search is performed case-insensitively.
- Verify** Verify disk. This option checks for a variety of problems, such as incorrect CRCs and mismatched main and backup data. This option does not automatically correct most problems, though; for that, you must use **gdisk**. If no problems are found, this command displays a summary of unallocated disk space.
- Write** Write data. Use this command to save your changes.

BUGS

Known bugs and limitations include:

- * The program compiles correctly only on Linux, FreeBSD, and Mac OS X. In theory, it should compile under Windows if the Ncurses library for Windows is installed, but I have not tested this capability. Linux versions for x86-64 (64-bit), x86 (32-bit), and PowerPC (32-bit) have been tested, with the x86-64 version having seen the most testing. Under FreeBSD, 32-bit (x86) and 64-bit (x86-64) versions have been tested. Only 32-bit versions for Mac OS X has been tested by the author.
- * The FreeBSD version of the program can't write changes to the partition table to a disk when existing partitions on that disk are mounted. (The same problem exists with many other FreeBSD utilities, such as **gpt**, **fdisk**, and **dd**.) This limitation can be overcome by typing **sysctl kern.geom.debugflags=16** at a shell prompt.
- * The program can load only up to 128 partitions (4 primary partitions and 124 logical partitions) when converting from MBR format. This limit can be raised by changing the `#define MAX_MBR_PARTS` line in the `basicmbr.h` source code file and recompiling; however, such a change will require using a larger-than-normal partition table. (The limit of 128 partitions was chosen because that number equals the 128 partitions supported by the most common partition table size.)
- * Converting from MBR format sometimes fails because of insufficient space at the start or (more commonly) the end of the disk. Resizing the partition table (using the 's' option in the experts' menu in **gdisk**) can sometimes overcome this problem; however, in extreme cases it may be

necessary to resize a partition using GNU Parted or a similar tool prior to conversion with GPT fdisk.

- * MBR conversions work only if the disk has correct LBA partition descriptors. These descriptors should be present on any disk over 8 GiB in size or on smaller disks partitioned with any but very ancient software.
- * BSD disklabel support can create first and/or last partitions that overlap with the GPT data structures. This can sometimes be compensated by adjusting the partition table size, but in extreme cases the affected partition(s) may need to be deleted.
- * Because of the highly variable nature of BSD disklabel structures, conversions from this form may be unreliable — partitions may be dropped, converted in a way that creates overlaps with other partitions, or converted with incorrect start or end values. Use this feature with caution!
- * Booting after converting an MBR or BSD disklabel disk is likely to be disrupted. Sometimes re-installing a boot loader will fix the problem, but other times you may need to switch boot loaders. Except on EFI-based platforms, Windows through at least Windows 7 doesn't support booting from GPT disks. Creating a hybrid MBR (using the 'h' option on the recovery & transformation menu in **gdisk**) or abandoning GPT in favor of MBR may be your only options in this case.
- * The **cgdisk** Verify function and the partition type listing obtainable by typing *L* in the Type function (or when specifying a partition type while creating a new partition) both currently exit ncurses mode. This limitation is a minor cosmetic blemish that does not affect functionality.

AUTHORS

Primary author: Roderick W. Smith (rodsbooks.com)

Contributors:

- * Yves Blusseau (1otnwmz02@sneakemail.com)
- * David Hubbard (david.c.hubbard@gmail.com)
- * Justin Maggard (justin.maggard@netgear.com)
- * Dwight Schauer (das@teegra.net)
- * Florian Zumbiehl (florz@florz.de)

SEE ALSO

cfdisk(8), **fdisk(8)**, **gdisk(8)**, **mkfs(8)**, **parted(8)**, **sfdisk(8)**, **sgdisk(8)**, **fixparts(8)**.

http://en.wikipedia.org/wiki/GUID_Partition_Table

<http://developer.apple.com/technotes/tn2006/tn2166.html>

<http://www.rodsbooks.com/gdisk/>

AVAILABILITY

The **cgdisk** command is part of the *GPTfdisk* package and is available from Rod Smith.

NAME

cgroups – Linux control groups

DESCRIPTION

Control groups, usually referred to as cgroups, are a Linux kernel feature which allow processes to be organized into hierarchical groups whose usage of various types of resources can then be limited and monitored. The kernel's cgroup interface is provided through a pseudo-filesystem called cgroupfs. Grouping is implemented in the core cgroup kernel code, while resource tracking and limits are implemented in a set of per-resource-type subsystems (memory, CPU, and so on).

Terminology

A *cgroup* is a collection of processes that are bound to a set of limits or parameters defined via the cgroup filesystem.

A *subsystem* is a kernel component that modifies the behavior of the processes in a cgroup. Various subsystems have been implemented, making it possible to do things such as limiting the amount of CPU time and memory available to a cgroup, accounting for the CPU time used by a cgroup, and freezing and resuming execution of the processes in a cgroup. Subsystems are sometimes also known as *resource controllers* (or simply, controllers).

The cgroups for a controller are arranged in a *hierarchy*. This hierarchy is defined by creating, removing, and renaming subdirectories within the cgroup filesystem. At each level of the hierarchy, attributes (e.g., limits) can be defined. The limits, control, and accounting provided by cgroups generally have effect throughout the subhierarchy underneath the cgroup where the attributes are defined. Thus, for example, the limits placed on a cgroup at a higher level in the hierarchy cannot be exceeded by descendant cgroups.

Cgroups version 1 and version 2

The initial release of the cgroups implementation was in Linux 2.6.24. Over time, various cgroup controllers have been added to allow the management of various types of resources. However, the development of these controllers was largely uncoordinated, with the result that many inconsistencies arose between controllers and management of the cgroup hierarchies became rather complex. A longer description of these problems can be found in the kernel source file *Documentation/admin-guide/cgroup-v2.rst* (or *Documentation/cgroup-v2.txt* in Linux 4.17 and earlier).

Because of the problems with the initial cgroups implementation (cgroups version 1), starting in Linux 3.10, work began on a new, orthogonal implementation to remedy these problems. Initially marked experimental, and hidden behind the *-o _DEVEL_sane_behavior* mount option, the new version (cgroups version 2) was eventually made official with the release of Linux 4.5. Differences between the two versions are described in the text below. The file *cgroup.sane_behavior*, present in cgroups v1, is a relic of this mount option. The file always reports "0" and is only retained for backward compatibility.

Although cgroups v2 is intended as a replacement for cgroups v1, the older system continues to exist (and for compatibility reasons is unlikely to be removed). Currently, cgroups v2 implements only a subset of the controllers available in cgroups v1. The two systems are implemented so that both v1 controllers and v2 controllers can be mounted on the same system. Thus, for example, it is possible to use those controllers that are supported under version 2, while also using version 1 controllers where version 2 does not yet support those controllers. The only restriction here is that a controller can't be simultaneously employed in both a cgroups v1 hierarchy and in the cgroups v2 hierarchy.

CGROUPS VERSION 1

Under cgroups v1, each controller may be mounted against a separate cgroup filesystem that provides its own hierarchical organization of the processes on the system. It is also possible to comount multiple (or even all) cgroups v1 controllers against the same cgroup filesystem, meaning that the comounted controllers manage the same hierarchical organization of processes.

For each mounted hierarchy, the directory tree mirrors the control group hierarchy. Each control group is represented by a directory, with each of its child control cgroups represented as a child directory. For instance, */user/joe/1.session* represents control group *1.session*, which is a child of cgroup *joe*, which is a child of */user*. Under each cgroup directory is a set of files which can be read or written to, reflecting resource limits and a few general cgroup properties.

Tasks (threads) versus processes

In cgroups v1, a distinction is drawn between *processes* and *tasks*. In this view, a process can consist of multiple tasks (more commonly called threads, from a user-space perspective, and called such in the remainder of this man page). In cgroups v1, it is possible to independently manipulate the cgroup memberships of the threads in a process.

The cgroups v1 ability to split threads across different cgroups caused problems in some cases. For example, it made no sense for the *memory* controller, since all of the threads of a process share a single address space. Because of these problems, the ability to independently manipulate the cgroup memberships of the threads in a process was removed in the initial cgroups v2 implementation, and subsequently restored in a more limited form (see the discussion of "thread mode" below).

Mounting v1 controllers

The use of cgroups requires a kernel built with the **CONFIG_CGROUP** option. In addition, each of the v1 controllers has an associated configuration option that must be set in order to employ that controller.

In order to use a v1 controller, it must be mounted against a cgroup filesystem. The usual place for such mounts is under a **tmpfs(5)** filesystem mounted at `/sys/fs/cgroup`. Thus, one might mount the *cpu* controller as follows:

```
mount -t cgroup -o cpu none /sys/fs/cgroup/cpu
```

It is possible to comount multiple controllers against the same hierarchy. For example, here the *cpu* and *cpuacct* controllers are comounted against a single hierarchy:

```
mount -t cgroup -o cpu,cpuacct none /sys/fs/cgroup/cpu,cpuacct
```

Comounting controllers has the effect that a process is in the same cgroup for all of the comounted controllers. Separately mounting controllers allows a process to be in cgroup `/foo1` for one controller while being in `/foo2/foo3` for another.

It is possible to comount all v1 controllers against the same hierarchy:

```
mount -t cgroup -o all cgroup /sys/fs/cgroup
```

(One can achieve the same result by omitting *-o all*, since it is the default if no controllers are explicitly specified.)

It is not possible to mount the same controller against multiple cgroup hierarchies. For example, it is not possible to mount both the *cpu* and *cpuacct* controllers against one hierarchy, and to mount the *cpu* controller alone against another hierarchy. It is possible to create multiple mount with exactly the same set of comounted controllers. However, in this case all that results is multiple mount points providing a view of the same hierarchy.

Note that on many systems, the v1 controllers are automatically mounted under `/sys/fs/cgroup`; in particular, **systemd(1)** automatically creates such mounts.

Unmounting v1 controllers

A mounted cgroup filesystem can be unmounted using the **umount(8)** command, as in the following example:

```
umount /sys/fs/cgroup/pids
```

But note well: a cgroup filesystem is unmounted only if it is not busy, that is, it has no child cgroups. If this is not the case, then the only effect of the **umount(8)** is to make the mount invisible. Thus, to ensure that the mount is really removed, one must first remove all child cgroups, which in turn can be done only after all member processes have been moved from those cgroups to the root cgroup.

Cgroups version 1 controllers

Each of the cgroups version 1 controllers is governed by a kernel configuration option (listed below). Additionally, the availability of the cgroups feature is governed by the **CONFIG_CGROUPS** kernel configuration option.

***cpu* (since Linux 2.6.24; **CONFIG_CGROUP_SCHED**)**

Cgroups can be guaranteed a minimum number of "CPU shares" when a system is busy. This does not limit a cgroup's CPU usage if the CPUs are not busy. For further information, see *Documentation/scheduler/sched-design-CFS.rst* (or *Documentation/scheduler/sched-design-CFS.txt* in Linux 5.2 and earlier).

In Linux 3.2, this controller was extended to provide CPU "bandwidth" control. If the kernel is configured with **CONFIG_CFS_BANDWIDTH**, then within each scheduling period (defined via a file in the cgroup directory), it is possible to define an upper limit on the CPU time allocated to the processes in a cgroup. This upper limit applies even if there is no other competition for the CPU. Further information can be found in the kernel source file *Documentation/scheduler/sched-bwc.rst* (or *Documentation/scheduler/sched-bwc.txt* in Linux 5.2 and earlier).

***cpuacct* (since Linux 2.6.24; **CONFIG_CGROUP_CPUACCT**)**

This provides accounting for CPU usage by groups of processes.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/cpuacct.rst* (or *Documentation/cgroup-v1/cpuacct.txt* in Linux 5.2 and earlier).

***cpuset* (since Linux 2.6.24; **CONFIG_CPUSETS**)**

This cgroup can be used to bind the processes in a cgroup to a specified set of CPUs and NUMA nodes.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/cpusets.rst* (or *Documentation/cgroup-v1/cpusets.txt* in Linux 5.2 and earlier).

***memory* (since Linux 2.6.25; **CONFIG_MEMCG**)**

The memory controller supports reporting and limiting of process memory, kernel memory, and swap used by cgroups.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/memory.rst* (or *Documentation/cgroup-v1/memory.txt* in Linux 5.2 and earlier).

***devices* (since Linux 2.6.26; **CONFIG_CGROUP_DEVICE**)**

This supports controlling which processes may create (mknod) devices as well as open them for reading or writing. The policies may be specified as allow-lists and deny-lists. Hierarchy is enforced, so new rules must not violate existing rules for the target or ancestor cgroups.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/devices.rst* (or *Documentation/cgroup-v1/devices.txt* in Linux 5.2 and earlier).

***freezer* (since Linux 2.6.28; **CONFIG_CGROUP_FREEZER**)**

The *freezer* cgroup can suspend and restore (resume) all processes in a cgroup. Freezing a cgroup /A also causes its children, for example, processes in /A/B, to be frozen.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/freezer-subsystem.rst* (or *Documentation/cgroup-v1/freezer-subsystem.txt* in Linux 5.2 and earlier).

***net_cls* (since Linux 2.6.29; **CONFIG_CGROUP_NET_CLASSID**)**

This places a classid, specified for the cgroup, on network packets created by a cgroup. These classids can then be used in firewall rules, as well as used to shape traffic using **tc(8)**. This applies only to packets leaving the cgroup, not to traffic arriving at the cgroup.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/net_cls.rst* (or *Documentation/cgroup-v1/net_cls.txt* in Linux 5.2 and earlier).

***blkio* (since Linux 2.6.33; **CONFIG_BLK_CGROUP**)**

The *blkio* cgroup controls and limits access to specified block devices by applying IO control in the form of throttling and upper limits against leaf nodes and intermediate nodes in the storage hierarchy.

Two policies are available. The first is a proportional-weight time-based division of disk implemented with CFQ. This is in effect for leaf nodes using CFQ. The second is a throttling policy which specifies upper I/O rate limits on a device.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/blkio-controller.rst* (or *Documentation/cgroup-v1/blkio-controller.txt* in Linux 5.2 and earlier).

***perf_event* (since Linux 2.6.39; **CONFIG_CGROUP_PERF**)**

This controller allows *perf* monitoring of the set of processes grouped in a cgroup.

Further information can be found in the kernel source files

***net_prio* (since Linux 3.3; **CONFIG_CGROUP_NET_PRIO**)**

This allows priorities to be specified, per network interface, for cgroups.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/net_prio.rst* (or *Documentation/cgroup-v1/net_prio.txt* in Linux 5.2 and earlier).

***hugetlb* (since Linux 3.5; **CONFIG_CGROUP_HUGETLB**)**

This supports limiting the use of huge pages by cgroups.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/hugetlb.rst* (or *Documentation/cgroup-v1/hugetlb.txt* in Linux 5.2 and earlier).

***pids* (since Linux 4.3; **CONFIG_CGROUP_PIDS**)**

This controller permits limiting the number of process that may be created in a cgroup (and its descendants).

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/pids.rst* (or *Documentation/cgroup-v1/pids.txt* in Linux 5.2 and earlier).

***rdma* (since Linux 4.11; **CONFIG_CGROUP_RDMA**)**

The RDMA controller permits limiting the use of RDMA/IB-specific resources per cgroup.

Further information can be found in the kernel source file *Documentation/admin-guide/cgroup-v1/rdma.rst* (or *Documentation/cgroup-v1/rdma.txt* in Linux 5.2 and earlier).

Creating cgroups and moving processes

A cgroup filesystem initially contains a single root cgroup, `'/'`, which all processes belong to. A new cgroup is created by creating a directory in the cgroup filesystem:

```
mkdir /sys/fs/cgroup/cpu/cg1
```

This creates a new empty cgroup.

A process may be moved to this cgroup by writing its PID into the cgroup's *cgroup.procs* file:

```
echo $$ > /sys/fs/cgroup/cpu/cg1/cgroup.procs
```

Only one PID at a time should be written to this file.

Writing the value 0 to a *cgroup.procs* file causes the writing process to be moved to the corresponding cgroup.

When writing a PID into the *cgroup.procs*, all threads in the process are moved into the new cgroup at once.

Within a hierarchy, a process can be a member of exactly one cgroup. Writing a process's PID to a *cgroup.procs* file automatically removes it from the cgroup of which it was previously a member.

The *cgroup.procs* file can be read to obtain a list of the processes that are members of a cgroup. The returned list of PIDs is not guaranteed to be in order. Nor is it guaranteed to be free of duplicates. (For example, a PID may be recycled while reading from the list.)

In cgroups v1, an individual thread can be moved to another cgroup by writing its thread ID (i.e., the kernel thread ID returned by **clone(2)** and **gettid(2)**) to the *tasks* file in a cgroup directory. This file can be read to discover the set of threads that are members of the cgroup.

Removing cgroups

To remove a cgroup, it must first have no child cgroups and contain no (nonzombie) processes. So long as that is the case, one can simply remove the corresponding directory pathname. Note that files in a cgroup directory cannot and need not be removed.

Cgroups v1 release notification

Two files can be used to determine whether the kernel provides notifications when a cgroup becomes empty. A cgroup is considered to be empty when it contains no child cgroups and no member processes.

A special file in the root directory of each cgroup hierarchy, *release_agent*, can be used to register the pathname of a program that may be invoked when a cgroup in the hierarchy becomes empty. The pathname of the newly empty cgroup (relative to the cgroup mount point) is provided as the sole command-line argument when the *release_agent* program is invoked. The *release_agent* program might remove the cgroup directory, or perhaps repopulate it with a process.

The default value of the *release_agent* file is empty, meaning that no release agent is invoked.

The content of the *release_agent* file can also be specified via a mount option when the cgroup filesystem is mounted:

```
mount -o release_agent=pathname ...
```

Whether or not the *release_agent* program is invoked when a particular cgroup becomes empty is determined by the value in the *notify_on_release* file in the corresponding cgroup directory. If this file contains the value 0, then the *release_agent* program is not invoked. If it contains the value 1, the *release_agent* program is invoked. The default value for this file in the root cgroup is 0. At the time when a new cgroup is created, the value in this file is inherited from the corresponding file in the parent cgroup.

Cgroup v1 named hierarchies

In cgroups v1, it is possible to mount a cgroup hierarchy that has no attached controllers:

```
mount -t cgroup -o none,name=somename none /some/mount/point
```

Multiple instances of such hierarchies can be mounted; each hierarchy must have a unique name. The only purpose of such hierarchies is to track processes. (See the discussion of release notification below.) An example of this is the *name=systemd* cgroup hierarchy that is used by **systemd(1)** to track services and user sessions.

Since Linux 5.0, the *cgroup_no_v1* kernel boot option (described below) can be used to disable cgroup v1 named hierarchies, by specifying *cgroup_no_v1=named*.

CGROUPS VERSION 2

In cgroups v2, all mounted controllers reside in a single unified hierarchy. While (different) controllers may be simultaneously mounted under the v1 and v2 hierarchies, it is not possible to mount the same controller simultaneously under both the v1 and the v2 hierarchies.

The new behaviors in cgroups v2 are summarized here, and in some cases elaborated in the following subsections.

- Cgroups v2 provides a unified hierarchy against which all controllers are mounted.
- "Internal" processes are not permitted. With the exception of the root cgroup, processes may reside only in leaf nodes (cgroups that do not themselves contain child cgroups). The details are somewhat more subtle than this, and are described below.
- Active cgroups must be specified via the files *cgroup.controllers* and *cgroup.subtree_control*.

- The *tasks* file has been removed. In addition, the *cgroup.oup.clone_children* file that is employed by the *cpuset* controller has been removed.
- An improved mechanism for notification of empty cgroups is provided by the *cgroup.events* file.

For more changes, see the *Documentation/admin-guide/cgroup-v2.rst* file in the kernel source (or *Documentation/cgroup-v2.txt* in Linux 4.17 and earlier).

Some of the new behaviors listed above saw subsequent modification with the addition in Linux 4.14 of "thread mode" (described below).

Cgroups v2 unified hierarchy

In cgroups v1, the ability to mount different controllers against different hierarchies was intended to allow great flexibility for application design. In practice, though, the flexibility turned out to be less useful than expected, and in many cases added complexity. Therefore, in cgroups v2, all available controllers are mounted against a single hierarchy. The available controllers are automatically mounted, meaning that it is not necessary (or possible) to specify the controllers when mounting the cgroup v2 filesystem using a command such as the following:

```
mount -t cgroup2 none /mnt/cgroup2
```

A cgroup v2 controller is available only if it is not currently in use via a mount against a cgroup v1 hierarchy. Or, to put things another way, it is not possible to employ the same controller against both a v1 hierarchy and the unified v2 hierarchy. This means that it may be necessary first to unmount a v1 controller (as described above) before that controller is available in v2. Since **systemd(1)** makes heavy use of some v1 controllers by default, it can in some cases be simpler to boot the system with selected v1 controllers disabled. To do this, specify the *cgroup_no_v1=list* option on the kernel boot command line; *list* is a comma-separated list of the names of the controllers to disable, or the word *all* to disable all v1 controllers. (This situation is correctly handled by **systemd(1)**, which falls back to operating without the specified controllers.)

Note that on many modern systems, **systemd(1)** automatically mounts the *cgroup2* filesystem at */sys/fs/cgroup/unified* during the boot process.

Cgroups v2 mount options

The following options (*mount -o*) can be specified when mounting the group v2 filesystem:

nsdelegate (since Linux 4.15)

Treat cgroup namespaces as delegation boundaries. For details, see below.

memory.localevents (since Linux 5.2)

The *memory.events* should show statistics only for the cgroup itself, and not for any descendant cgroups. This was the behavior before Linux 5.2. Starting in Linux 5.2, the default behavior is to include statistics for descendant cgroups in *memory.events*, and this mount option can be used to revert to the legacy behavior. This option is system wide and can be set on mount or modified through remount only from the initial mount namespace; it is silently ignored in noninitial namespaces.

Cgroups v2 controllers

The following controllers, documented in the kernel source file *Documentation/admin-guide/cgroup-v2.rst* (or *Documentation/cgroup-v2.txt* in Linux 4.17 and earlier), are supported in cgroups version 2:

cpu (since Linux 4.15)

This is the successor to the version 1 *cpu* and *cpuacct* controllers.

cpuset (since Linux 5.0)

This is the successor of the version 1 *cpuset* controller.

freezer (since Linux 5.2)

This is the successor of the version 1 *freezer* controller.

hugetlb (since Linux 5.6)

This is the successor of the version 1 *hugetlb* controller.

io (since Linux 4.5)

This is the successor of the version 1 *blkio* controller.

memory (since Linux 4.5)

This is the successor of the version 1 *memory* controller.

perf_event (since Linux 4.11)

This is the same as the version 1 *perf_event* controller.

pids (since Linux 4.5)

This is the same as the version 1 *pids* controller.

rdma (since Linux 4.11)

This is the same as the version 1 *rdma* controller.

There is no direct equivalent of the *net_cls* and *net_prio* controllers from cgroups version 1. Instead, support has been added to **iptables(8)** to allow eBPF filters that hook on cgroup v2 pathnames to make decisions about network traffic on a per-cgroup basis.

The v2 *devices* controller provides no interface files; instead, device control is gated by attaching an eBPF (**BPF_CGROUP_DEVICE**) program to a v2 cgroup.

Cgroups v2 subtree control

Each cgroup in the v2 hierarchy contains the following two files:

cgroup.controllers

This read-only file exposes a list of the controllers that are *available* in this cgroup. The contents of this file match the contents of the *cgroup.subtree_control* file in the parent cgroup.

cgroup.subtree_control

This is a list of controllers that are *active (enabled)* in the cgroup. The set of controllers in this file is a subset of the set in the *cgroup.controllers* of this cgroup. The set of active controllers is modified by writing strings to this file containing space-delimited controller names, each preceded by '+' (to enable a controller) or '-' (to disable a controller), as in the following example:

```
echo '+pids -memory' > x/y/cgroup.subtree_control
```

An attempt to enable a controller that is not present in *cgroup.controllers* leads to an **ENOENT** error when writing to the *cgroup.subtree_control* file.

Because the list of controllers in *cgroup.subtree_control* is a subset of those *cgroup.controllers*, a controller that has been disabled in one cgroup in the hierarchy can never be re-enabled in the subtree below that cgroup.

A cgroup's *cgroup.subtree_control* file determines the set of controllers that are exercised in the *child* cgroups. When a controller (e.g., *pids*) is present in the *cgroup.subtree_control* file of a parent cgroup, then the corresponding controller-interface files (e.g., *pids.max*) are automatically created in the children of that cgroup and can be used to exert resource control in the child cgroups.

Cgroups v2 "no internal processes" rule

Cgroups v2 enforces a so-called "no internal processes" rule. Roughly speaking, this rule means that, with the exception of the root cgroup, processes may reside only in leaf nodes (cgroups that do not themselves contain child cgroups). This avoids the need to decide how to partition resources between processes which are members of cgroup A and processes in child cgroups of A.

For instance, if cgroup */cg1/cg2* exists, then a process may reside in */cg1/cg2*, but not in */cg1*. This is to avoid an ambiguity in cgroups v1 with respect to the delegation of resources between processes in */cg1* and its child cgroups. The recommended approach in cgroups v2 is to create a subdirectory called *leaf* for any nonleaf cgroup which should contain processes, but no child cgroups. Thus, processes which previously would have gone into */cg1* would now go into */cg1/leaf*. This has the advantage of making explicit the relationship between processes in */cg1/leaf* and */cg1*'s other children.

The "no internal processes" rule is in fact more subtle than stated above. More precisely, the rule is that a (nonroot) cgroup can't both (1) have member processes, and (2) distribute resources into child cgroups—that is, have a nonempty *cgroup.subtree_control* file. Thus, it is possible for a cgroup to have both member processes and child cgroups, but before controllers can be enabled for that cgroup, the member processes must be moved out of the cgroup (e.g., perhaps into the child cgroups).

With the Linux 4.14 addition of "thread mode" (described below), the "no internal processes" rule has been relaxed in some cases.

Cgroups v2 cgroup.events file

Each nonroot cgroup in the v2 hierarchy contains a read-only file, *cgroup.events*, whose contents are key-value pairs (delimited by newline characters, with the key and value separated by spaces) providing state information about the cgroup:

```
$ cat mygrp/cgroup.events
populated 1
frozen 0
```

The following keys may appear in this file:

populated

The value of this key is either 1, if this cgroup or any of its descendants has member processes, or otherwise 0.

frozen (since Linux 5.2)

The value of this key is 1 if this cgroup is currently frozen, or 0 if it is not.

The *cgroup.events* file can be monitored, in order to receive notification when the value of one of its keys changes. Such monitoring can be done using *inotify(7)*, which notifies changes as **IN_MODIFY** events, or *poll(2)*, which notifies changes by returning the **POLLPRI** and **POLLERR** bits in the *revents* field.

Cgroup v2 release notification

Cgroups v2 provides a new mechanism for obtaining notification when a cgroup becomes empty. The cgroups v1 *release_agent* and *notify_on_release* files are removed, and replaced by the *populated* key in the *cgroup.events* file. This key either has the value 0, meaning that the cgroup (and its descendants) contain no (nonzombie) member processes, or 1, meaning that the cgroup (or one of its descendants) contains member processes.

The cgroups v2 release-notification mechanism offers the following advantages over the cgroups v1 *release_agent* mechanism:

- It allows for cheaper notification, since a single process can monitor multiple *cgroup.events* files (using the techniques described earlier). By contrast, the cgroups v1 mechanism requires the expense of creating a process for each notification.
- Notification for different cgroup subhierarchies can be delegated to different processes. By contrast, the cgroups v1 mechanism allows only one release agent for an entire hierarchy.

Cgroups v2 cgroup.stat file

Each cgroup in the v2 hierarchy contains a read-only *cgroup.stat* file (first introduced in Linux 4.14) that consists of lines containing key-value pairs. The following keys currently appear in this file:

nr_descendants

This is the total number of visible (i.e., living) descendant cgroups underneath this cgroup.

nr_dying_descendants

This is the total number of dying descendant cgroups underneath this cgroup. A cgroup enters the dying state after being deleted. It remains in that state for an undefined period (which will depend on system load) while resources are freed before the cgroup is destroyed. Note that the presence of some cgroups in the dying state is normal, and is not indicative of any problem.

A process can't be made a member of a dying cgroup, and a dying cgroup can't be brought back to life.

Limiting the number of descendant cgroups

Each cgroup in the v2 hierarchy contains the following files, which can be used to view and set limits on the number of descendant cgroups under that cgroup:

cgroup.max.depth (since Linux 4.14)

This file defines a limit on the depth of nesting of descendant cgroups. A value of 0 in this file means that no descendant cgroups can be created. An attempt to create a descendant whose nesting level exceeds the limit fails (*mkdir(2)* fails with the error **EAGAIN**).

Writing the string "max" to this file means that no limit is imposed. The default value in this file is "max".

cgroup.max.descendants (since Linux 4.14)

This file defines a limit on the number of live descendant cgroups that this cgroup may have. An attempt to create more descendants than allowed by the limit fails (*mkdir(2)* fails with the error **EAGAIN**).

Writing the string "max" to this file means that no limit is imposed. The default value in this file is "max".

CGROUPS DELEGATION: DELEGATING A HIERARCHY TO A LESS PRIVILEGED USER

In the context of cgroups, delegation means passing management of some subtree of the cgroup hierarchy to a nonprivileged user. Cgroups v1 provides support for delegation based on file permissions in the cgroup hierarchy but with less strict containment rules than v2 (as noted below). Cgroups v2 supports delegation with containment by explicit design. The focus of the discussion in this section is on delegation in cgroups v2, with some differences for cgroups v1 noted along the way.

Some terminology is required in order to describe delegation. *Adele gater* is a privileged user (i.e., root) who owns a parent cgroup. A *delegatee* is a nonprivileged user who will be granted the permissions needed to manage some subhierarchy under that parent cgroup, known as the *delegated subtree*.

To perform delegation, the delegator makes certain directories and files writable by the delegatee, typically by changing the ownership of the objects to be the user ID of the delegatee. Assuming that we want to delegate the hierarchy rooted at (say) */dlgt_grp* and that there are not yet any child cgroups under that cgroup, the ownership of the following is changed to the user ID of the delegatee:

/dlgt_grp

Changing the ownership of the root of the subtree means that any new cgroups created under the subtree (and the files they contain) will also be owned by the delegatee.

/dlgt_grp/cgroup.procs

Changing the ownership of this file means that the delegatee can move processes into the root of the delegated subtree.

/dlgt_grp/cgroup.subtree_control (cgroups v2 only)

Changing the ownership of this file means that the delegatee can enable controllers (that are present in */dlgt_grp/cgroup.controllers*) in order to further redistribute resources at lower levels in the subtree. (As an alternative to changing the ownership of this file, the delegator might instead add selected controllers to this file.)

/dlgt_grp/cgroup.threads (cgroups v2 only)

Changing the ownership of this file is necessary if a threaded subtree is being delegated (see the description of "thread mode", below). This permits the delegatee to write thread IDs to the file. (The ownership of this file can also be changed when delegating a domain subtree, but currently this serves no purpose, since, as described below, it is not possible to move a thread between domain cgroups by writing its thread ID to the *cgroup.threads* file.)

In cgroups v1, the corresponding file that should instead be delegated is the *tasks* file.

The delegator should *not* change the ownership of any of the controller interfaces files (e.g., *pids.max*, *memory.high*) in *dlgt_grp*. Those files are used from the next level above the delegated subtree in order to distribute resources into the subtree, and the delegatee should not have permission to change the resources

that are distributed into the delegated subtree.

See also the discussion of the */sys/kernel/cgroup/delegate* file in NOTES for information about further delegatable files in cgroups v2.

After the aforementioned steps have been performed, the delegatee can create child cgroups within the delegated subtree (the cgroup subdirectories and the files they contain will be owned by the delegatee) and move processes between cgroups in the subtree. If some controllers are present in *dlgt_grp/cgroup.subtree_control*, or the ownership of that file was passed to the delegatee, the delegatee can also control the further redistribution of the corresponding resources into the delegated subtree.

Cgroups v2 delegation: nsdelegate and cgroup namespaces

Starting with Linux 4.13, there is a second way to perform cgroup delegation in the cgroups v2 hierarchy. This is done by mounting or remounting the cgroup v2 filesystem with the *nsdelegate* mount option. For example, if the cgroup v2 filesystem has already been mounted, we can remount it with the *nsdelegate* option as follows:

```
mount -t cgroup2 -o remount,nsdelegate \
    none /sys/fs/cgroup/unified
```

The effect of this mount option is to cause cgroup namespaces to automatically become delegation boundaries. More specifically, the following restrictions apply for processes inside the cgroup namespace:

- Writes to controller interface files in the root directory of the namespace will fail with the error **EPERM**. Processes inside the cgroup namespace can still write to delegatable files in the root directory of the cgroup namespace such as *cgroup.procs* and *cgroup.subtree_control*, and can create subhierarchy underneath the root directory.
- Attempts to migrate processes across the namespace boundary are denied (with the error **ENOENT**). Processes inside the cgroup namespace can still (subject to the containment rules described below) move processes between cgroups *within* the subhierarchy under the namespace root.

The ability to define cgroup namespaces as delegation boundaries makes cgroup namespaces more useful. To understand why, suppose that we already have one cgroup hierarchy that has been delegated to a non-privileged user, *cecilia*, using the older delegation technique described above. Suppose further that *cecilia* wanted to further delegate a subhierarchy under the existing delegated hierarchy. (For example, the delegated hierarchy might be associated with an unprivileged container run by *cecilia*.) Even if a cgroup namespace was employed, because both hierarchies are owned by the unprivileged user *cecilia*, the following illegitimate actions could be performed:

- A process in the inferior hierarchy could change the resource controller settings in the root directory of that hierarchy. (These resource controller settings are intended to allow control to be exercised from the *parent* cgroup; a process inside the child cgroup should not be allowed to modify them.)
- A process inside the inferior hierarchy could move processes into and out of the inferior hierarchy if the cgroups in the superior hierarchy were somehow visible.

Employing the *nsdelegate* mount option prevents both of these possibilities.

The *nsdelegate* mount option only has an effect when performed in the initial mount namespace; in other mount namespaces, the option is silently ignored.

Note: On some systems, **systemd(1)** automatically mounts the cgroup v2 filesystem. In order to experiment with the *nsdelegate* operation, it may be useful to boot the kernel with the following command-line options:

```
cgroup_no_v1=all systemd.legacy_systemd_cgroup_controller
```

These options cause the kernel to boot with the cgroups v1 controllers disabled (meaning that the controllers are available in the v2 hierarchy), and tells **systemd(1)** not to mount and use the cgroup v2 hierarchy, so that the v2 hierarchy can be manually mounted with the desired options after boot-up.

Cgroup delegation containment rules

Some delegation *containment rules* ensure that the delegatee can move processes between cgroups within the delegated subtree, but can't move processes from outside the delegated subtree into the subtree or vice versa. A nonprivileged process (i.e., the delegatee) can write the PID of a "target" process into a *cgroup.procs* file only if all of the following are true:

- The writer has write permission on the *cgroup.procs* file in the destination cgroup.
- The writer has write permission on the *cgroup.procs* file in the nearest common ancestor of the source and destination cgroups. Note that in some cases, the nearest common ancestor may be the source or destination cgroup itself. This requirement is not enforced for cgroups v1 hierarchies, with the consequence that containment in v1 is less strict than in v2. (For example, in cgroups v1 the user that owns two distinct delegated subhierarchies can move a process between the hierarchies.)
- If the cgroup v2 filesystem was mounted with the *nsdelegate* option, the writer must be able to see the source and destination cgroups from its cgroup namespace.
- In cgroups v1: the effective UID of the writer (i.e., the delegatee) matches the real user ID or the saved set-user-ID of the target process. Before Linux 4.11, this requirement also applied in cgroups v2 (This was a historical requirement inherited from cgroups v1 that was later deemed unnecessary, since the other rules suffice for containment in cgroups v2.)

Note: one consequence of these delegation containment rules is that the unprivileged delegatee can't place the first process into the delegated subtree; instead, the delegater must place the first process (a process owned by the delegatee) into the delegated subtree.

CGROUPS VERSION 2 THREAD MODE

Among the restrictions imposed by cgroups v2 that were not present in cgroups v1 are the following:

- *No thread-granularity control:* all of the threads of a process must be in the same cgroup.
- *No internal processes:* a cgroup can't both have member processes and exercise controllers on child cgroups.

Both of these restrictions were added because the lack of these restrictions had caused problems in cgroups v1. In particular, the cgroups v1 ability to allow thread-level granularity for cgroup membership made no sense for some controllers. (A notable example was the *memory* controller: since threads share an address space, it made no sense to split threads across different *memory* cgroups.)

Notwithstanding the initial design decision in cgroups v2, there were use cases for certain controllers, notably the *cpu* controller, for which thread-level granularity of control was meaningful and useful. To accommodate such use cases, Linux 4.14 added *thread mode* for cgroups v2.

Thread mode allows the following:

- The creation of *threaded subtrees* in which the threads of a process may be spread across cgroups inside the tree. (A threaded subtree may contain multiple multithreaded processes.)
- The concept of *threaded controllers*, which can distribute resources across the cgroups in a threaded subtree.
- A relaxation of the "no internal processes rule", so that, within a threaded subtree, a cgroup can both contain member threads and exercise resource control over child cgroups.

With the addition of thread mode, each nonroot cgroup now contains a new file, *cgroup.type*, that exposes, and in some circumstances can be used to change, the "type" of a cgroup. This file contains one of the following type values:

domain This is a normal v2 cgroup that provides process-granularity control. If a process is a member of this cgroup, then all threads of the process are (by definition) in the same cgroup. This is the default cgroup type, and provides the same behavior that was provided for cgroups in the initial cgroups v2 implementation.

threaded

This cgroup is a member of a threaded subtree. Threads can be added to this cgroup, and controllers can be enabled for the cgroup.

domain threaded

This is a domain cgroup that serves as the root of a threaded subtree. This cgroup type is also known as "threaded root".

domain invalid

This is a cgroup inside a threaded subtree that is in an "invalid" state. Processes can't be added to the cgroup, and controllers can't be enabled for the cgroup. The only thing that can be done with this cgroup (other than deleting it) is to convert it to a *threaded* cgroup by writing the string "*threaded*" to the *cgroup.type* file.

The rationale for the existence of this "interim" type during the creation of a threaded subtree (rather than the kernel simply immediately converting all cgroups under the threaded root to the type *threaded*) is to allow for possible future extensions to the thread mode model

Threaded versus domain controllers

With the addition of threads mode, cgroups v2 now distinguishes two types of resource controllers:

- *Threaded* controllers: these controllers support thread-granularity for resource control and can be enabled inside threaded subtrees, with the result that the corresponding controller-interface files appear inside the cgroups in the threaded subtree. As at Linux 4.19, the following controllers are threaded: *cpu*, *perf_event*, and *pids*.
- *Domain* controllers: these controllers support only process granularity for resource control. From the perspective of a domain controller, all threads of a process are always in the same cgroup. Domain controllers can't be enabled inside a threaded subtree.

Creating a threaded subtree

There are two pathways that lead to the creation of a threaded subtree. The first pathway proceeds as follows:

- (1) We write the string "*threaded*" to the *cgroup.type* file of a cgroup *y/z* that currently has the type *domain*. This has the following effects:
 - The type of the cgroup *y/z* becomes *threaded*.
 - The type of the parent cgroup, *y*, becomes *domain threaded*. The parent cgroup is the root of a threaded subtree (also known as the "threaded root").
 - All other cgroups under *y* that were not already of type *threaded* (because they were inside already existing threaded subtrees under the new threaded root) are converted to type *domain invalid*. Any subsequently created cgroups under *y* will also have the type *domain invalid*.
- (2) We write the string "*threaded*" to each of the *domain invalid* cgroups under *y*, in order to convert them to the type *threaded*. As a consequence of this step, all threads under the threaded root now have the type *threaded* and the threaded subtree is now fully usable. The requirement to write "*threaded*" to each of these cgroups is somewhat cumbersome, but allows for possible future extensions to the thread-mode model.

The second way of creating a threaded subtree is as follows:

- (1) In an existing cgroup, *z*, that currently has the type *domain*, we (1.1) enable one or more threaded controllers and (1.2) make a process a member of *z*. (These two steps can be done in either order.) This has the following consequences:
 - The type of *z* becomes *domain threaded*.
 - All of the descendant cgroups of *x* that were not already of type *threaded* are converted to type *domain invalid*.
- (2) As before, we make the threaded subtree usable by writing the string "*threaded*" to each of the *domain invalid* cgroups under *y*, in order to convert them to the type *threaded*.

One of the consequences of the above pathways to creating a threaded subtree is that the threaded root cgroup can be a parent only to *threaded* (and *domain invalid*) cgroups. The threaded root cgroup can't be a parent of a *domain* cgroups, and a *threaded* cgroup can't have a sibling that is a *domain* cgroup.

Using a threaded subtree

Within a threaded subtree, threaded controllers can be enabled in each subgroup whose type has been changed to *threaded*; upon doing so, the corresponding controller interface files appear in the children of that cgroup.

A process can be moved into a threaded subtree by writing its PID to the *cgroup.procs* file in one of the cgroups inside the tree. This has the effect of making all of the threads in the process members of the corresponding cgroup and makes the process a member of the threaded subtree. The threads of the process can then be spread across the threaded subtree by writing their thread IDs (see **gettid(2)**) to the *cgroup.threads* files in different cgroups inside the subtree. The threads of a process must all reside in the same threaded subtree.

As with writing to *cgroup.procs*, some containment rules apply when writing to the *cgroup.threads* file:

- The writer must have write permission on the *cgroup.threads* file in the destination cgroup.
- The writer must have write permission on the *cgroup.procs* file in the common ancestor of the source and destination cgroups. (In some cases, the common ancestor may be the source or destination cgroup itself.)
- The source and destination cgroups must be in the same threaded subtree. (Outside a threaded subtree, an attempt to move a thread by writing its thread ID to the *cgroup.threads* file in a different *domain* cgroup fails with the error **EOPNOTSUPP**.)

The *cgroup.threads* file is present in each cgroup (including *domain* cgroups) and can be read in order to discover the set of threads that is present in the cgroup. The set of thread IDs obtained when reading this file is not guaranteed to be ordered or free of duplicates.

The *cgroup.procs* file in the threaded root shows the PIDs of all processes that are members of the threaded subtree. The *cgroup.procs* files in the other cgroups in the subtree are not readable.

Domain controllers can't be enabled in a threaded subtree; no controller-interface files appear inside the cgroups underneath the threaded root. From the point of view of a domain controller, threaded subtrees are invisible: a multithreaded process inside a threaded subtree appears to a domain controller as a process that resides in the threaded root cgroup.

Within a threaded subtree, the "no internal processes" rule does not apply: a cgroup can both contain member processes (or thread) and exercise controllers on child cgroups.

Rules for writing to *cgroup.type* and creating threaded subtrees

A number of rules apply when writing to the *cgroup.type* file:

- Only the string "*threaded*" may be written. In other words, the only explicit transition that is possible is to convert a *domain* cgroup to type *threaded*.
- The effect of writing "*threaded*" depends on the current value in *cgroup.type*, as follows:
 - *domain* or *domain threaded*: start the creation of a threaded subtree (whose root is the parent of this cgroup) via the first of the pathways described above;
 - *domain invalid*: convert this cgroup (which is inside a threaded subtree) to a usable (i.e., *threaded*) state;
 - *threaded*: no effect (a "no-op").
- We can't write to a *cgroup.type* file if the parent's type is *domain invalid*. In other words, the cgroups of a threaded subtree must be converted to the *threaded* state in a top-down manner.

There are also some constraints that must be satisfied in order to create a threaded subtree rooted at the cgroup *x*:

- There can be no member processes in the descendant cgroups of *x*. (The *cgroupx* can itself have member processes.)
- No domain controllers may be enabled in *x*'s *cgroup.subtree_control* file.

If any of the above constraints is violated, then an attempt to write "*threaded*" to a *cgroup.type* file fails with the error **ENOTSUP**.

The "domain threaded" cgroup type

According to the pathways described above, the type of a cgroup can change to *domain threaded* in either of the following cases:

- The string "*threaded*" is written to a child cgroup.
- A threaded controller is enabled inside the cgroup and a process is made a member of the cgroup.

A *domain threaded* cgroup, *x*, can revert to the type *domain* if the above conditions no longer hold true—that is, if all *threaded* child cgroups of *x* are removed and either *x* no longer has threaded controllers enabled or no longer has member processes.

When a *domain threaded* cgroup *x* reverts to the type *domain*:

- All *domain invalid* descendants of *x* that are not in lower-level threaded subtrees revert to the type *domain*.
- The root cgroups in any lower-level threaded subtrees revert to the type *domain threaded*.

Exceptions for the root cgroup

The root cgroup of the v2 hierarchy is treated exceptionally: it can be the parent of both *domain* and *threaded* cgroups. If the string "*threaded*" is written to the *cgroup.type* file of one of the children of the root cgroup, then

- The type of that cgroup becomes *threaded*.
- The type of any descendants of that cgroup that are not part of lower-level threaded subtrees changes to *domain invalid*.

Note that in this case, there is no cgroup whose type becomes *domain threaded*. (Notionally, the root cgroup can be considered as the threaded root for the cgroup whose type was changed to *threaded*.)

The aim of this exceptional treatment for the root cgroup is to allow a threaded cgroup that employs the *cpu* controller to be placed as high as possible in the hierarchy, so as to minimize the (small) cost of traversing the cgroup hierarchy.

The cgroups v2 "cpu" controller and realtime threads

As at Linux 4.19, the cgroups v2 *cpu* controller does not support control of realtime threads (specifically threads scheduled under any of the policies **SCHED_FIFO**, **SCHED_RR**, described **SCHED_DEADLINE**; see **sched(7)**). Therefore, the *cpu* controller can be enabled in the root cgroup only if all realtime threads are in the root cgroup. (If there are realtime threads in nonroot cgroups, then a **write(2)** of the string "+*cpu*" to the *cgroup.subtree_control* file fails with the error **EINVAL**.)

On some systems, **systemd(1)** places certain realtime threads in nonroot cgroups in the v2 hierarchy. On such systems, these threads must first be moved to the root cgroup before the *cpu* controller can be enabled.

ERRORS

The following errors can occur for **mount(2)**:

EBUSY

An attempt to mount a cgroup version 1 filesystem specified neither the *name=* option (to mount a named hierarchy) nor a controller name (or *all*).

NOTES

A child process created via **fork(2)** inherits its parent's cgroup memberships. A process's cgroup memberships are preserved across **execve(2)**.

The **clone3(2)** **CLONE_INTO_CGROUP** flag can be used to create a child process that begins its life in a

different version 2 cgroup from the parent process.

/proc files

/proc/cgroups (since Linux 2.6.24)

This file contains information about the controllers that are compiled into the kernel. An example of the contents of this file (reformatted for readability) is the following:

#subsys_name	hierarchy	num_cgroups	enabled
cpuset	4	1	1
cpu	8	1	1
cpuacct	8	1	1
blkio	6	1	1
memory	3	1	1
devices	10	84	1
freezer	7	1	1
net_cls	9	1	1
perf_event	5	1	1
net_prio	9	1	1
hugetlb	0	1	0
pids	2	1	1

The fields in this file are, from left to right:

- [1] The name of the controller.
- [2] The unique ID of the cgroup hierarchy on which this controller is mounted. If multiple cgroups v1 controllers are bound to the same hierarchy, then each will show the same hierarchy ID in this field. The value in this field will be 0 if:
 - the controller is not mounted on a cgroups v1 hierarchy;
 - the controller is bound to the cgroups v2 single unified hierarchy; or
 - the controller is disabled (see below).
- [3] The number of control groups in this hierarchy using this controller.
- [4] This field contains the value 1 if this controller is enabled, or 0 if it has been disabled (via the *cgroup_disable* kernel command-line boot parameter).

/proc/[pid]/cgroup (since Linux 2.6.24)

This file describes control groups to which the process with the corresponding PID belongs. The displayed information differs for cgroups version 1 and version 2 hierarchies.

For each cgroup hierarchy of which the process is a member, there is one entry containing three colon-separated fields:

hierarchy-ID:controller-list:cgroup-path

For example:

5:cpuacct,cpu,cpuset:/daemons

The colon-separated fields are, from left to right:

- [1] For cgroups version 1 hierarchies, this field contains a unique hierarchy ID number that can be matched to a hierarchy ID in /proc/cgroups. For the cgroups version 2 hierarchy, this field contains the value 0.
- [2] For cgroups version 1 hierarchies, this field contains a comma-separated list of the controllers bound to the hierarchy. For the cgroups version 2 hierarchy, this field is empty.
- [3] This field contains the pathname of the control group in the hierarchy to which the process belongs. This pathname is relative to the mount point of the hierarchy.

/sys/kernel/cgroup files*/sys/kernel/cgroup/delegate* (since Linux 4.15)

This file exports a list of the cgroups v2 files (one per line) that are delegatable (i.e., whose ownership should be changed to the user ID of the delegatee). In the future, the set of delegatable files may change or grow, and this file provides a way for the kernel to inform user-space applications of which files must be delegated. As at Linux 4.15, one sees the following when inspecting this file:

```
$ cat /sys/kernel/cgroup/delegate
cgroup.procs
cgroup.subtree_control
cgroup.threads
```

/sys/kernel/cgroup/features (since Linux 4.15)

Over time, the set of cgroups v2 features that are provided by the kernel may change or grow, or some features may not be enabled by default. This file provides a way for user-space applications to discover what features the running kernel supports and has enabled. Features are listed one per line:

```
$ cat /sys/kernel/cgroup/features
nsdelegate
memory_localevents
```

The entries that can appear in this file are:

memory_localevents (since Linux 5.2)

The kernel supports the *memory_localevents* mount option.

nsdelegate (since Linux 4.15)

The kernel supports the *nsdelegate* mount option.

memory_recursiveprot (since Linux 5.7)

The kernel supports the *memory_recursiveprot* mount option.

SEE ALSO

**prlimit(1), systemd(1), systemd-cgls(1), systemd-cgtop(1), clone(2), ioprio_set(2),
perf_event_open(2), setrlimit(2), cgroup_namespaces(7), cpuset(7), namespaces(7), sched(7),
user_namespaces(7)**

The kernel source file *Documentation/admin-guide/cgroup-v2.rst*.

NAME

chage – change user password expiry information

SYNOPSIS

chage [*options*] *LOGIN*

DESCRIPTION

The **chage** command changes the number of days between password changes and the date of the last password change. This information is used by the system to determine when a user must change their password.

OPTIONS

The options which apply to the **chage** command are:

-d, --lastday *LAST_DAY*

Set the number of days since January 1st, 1970 when the password was last changed. The date may also be expressed in the format YYYY-MM-DD (or the format more commonly used in your area).

-E, --expiredate *EXPIRE_DATE*

Set the date or number of days since January 1, 1970 on which the user's account will no longer be accessible. The date may also be expressed in the format YYYY-MM-DD (or the format more commonly used in your area). A user whose account is locked must contact the system administrator before being able to use the system again.

Passing the number *-I* as the *EXPIRE_DATE* will remove an account expiration date.

-h, --help

Display help message and exit.

-i, --iso8601

When printing dates, use YYYY-MM-DD format.

-I, --inactive *INACTIVE*

Set the number of days of inactivity after a password has expired before the account is locked. The *INACTIVE* option is the number of days of inactivity. A user whose account is locked must contact the system administrator before being able to use the system again.

Passing the number *-I* as the *INACTIVE* will remove an account's inactivity.

-l, --list

Show account aging information.

-m, --mindays *MIN_DAYS*

Set the minimum number of days between password changes to *MIN_DAYS*. A value of zero for this field indicates that the user may change their password at any time.

-M, --maxdays *MAX_DAYS*

Set the maximum number of days during which a password is valid. When *MAX_DAYS* plus *LAST_DAY* is less than the current day, the user will be required to change their password before being able to use their account. This occurrence can be planned for in advance by use of the **-W** option, which provides the user with advance warning.

Passing the number *-I* as *MAX_DAYS* will remove checking a password's validity.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-W, --warndays *WARN_DAYS*

Set the number of days of warning before a password change is required. The *WARN_DAYS* option is the number of days prior to the password expiring that a user will be warned their password is about to expire.

If none of the options are selected, **chage** operates in an interactive fashion, prompting the user with the current values for all of the fields. Enter the new value to change the field, or leave the line blank to use the current value. The current value is displayed between a pair of [] marks.

NOTE

The **chage** program requires a shadow password file to be available.

The **chage** command is restricted to the root user, except for the **-l** option, which may be used by an unprivileged user to determine when their password or account is due to expire.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

FILES

/etc/passwd
User account information.

/etc/shadow
Secure user account information.

EXIT VALUES

The **chage** command exits with the following values:

0	success
1	permission denied
2	invalid command syntax
15	can't find the shadow password file

SEE ALSO

passwd(5), shadow(5).

NAME

chgpasswd – update group passwords in batch mode

SYNOPSIS

chgpasswd [*options*]

DESCRIPTION

The **chgpasswd** command reads a list of group name and password pairs from standard input and uses this information to update a set of existing groups. Each line is of the format:

group_name:password

By default the supplied password must be in clear-text, and is encrypted by **chgpasswd**.

The default encryption algorithm can be defined for the system with the **ENCRYPT_METHOD** variable of /etc/login.defs, and can be overwritten with the **-e**, **-m**, or **-c** options.

This command is intended to be used in a large system environment where many accounts are created at a single time.

OPTIONS

The options which apply to the **chgpasswd** command are:

-c, --crypt-method

Use the specified method to encrypt the passwords.

The available methods are DES, MD5, NONE, and SHA256 or SHA512 if your libc support these methods.

-e, --encrypted

Supplied passwords are in encrypted form.

-h, --help

Display help message and exit.

-m, --md5

Use MD5 encryption instead of DES when the supplied passwords are not encrypted.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-s, --sha-rounds

Use the specified number of rounds to encrypt the passwords.

The value 0 means that the system will choose the default number of rounds for the crypt method (5000).

A minimal value of 1000 and a maximal value of 999,999,999 will be enforced.

You can only use this option with the SHA256 or SHA512 crypt method.

By default, the number of rounds is defined by the **SHA_CRYPT_MIN_ROUNDS** and **SHA_CRYPT_MAX_ROUNDS** variables in /etc/login.defs.

CAVEATS

Remember to set permissions or umask to prevent readability of unencrypted files by other users.

You should make sure the passwords and the encryption method respect the system's password policy.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

ENCRYPT_METHOD (string)

This defines the system default encryption algorithm for encrypting passwords (if no algorithm are

specified on the command line).

It can take one of these values: *DES* (default), *MD5*, *SHA256*, *SHA512*.

Note: this parameter overrides the **MD5_CRYPT_ENAB** variable.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

MD5_CRYPT_ENAB (boolean)

Indicate if passwords must be encrypted using the MD5-based algorithm. If set to *yes*, new passwords will be encrypted using the MD5-based algorithm compatible with the one used by recent releases of FreeBSD. It supports passwords of unlimited length and longer salt strings. Set to *no* if you need to copy encrypted passwords to other systems which don't understand the new algorithm. Default is *no*.

This variable is superseded by the **ENCRYPT_METHOD** variable or by any command line option used to configure the encryption algorithm.

This variable is deprecated. You should use **ENCRYPT_METHOD**.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

SHA_CRYPT_MIN_ROUNDS (number), **SHA_CRYPT_MAX_ROUNDS** (number)

When **ENCRYPT_METHOD** is set to *SHA256* or *SHA512*, this defines the number of SHA rounds used by the encryption algorithm by default (when the number of rounds is not specified on the command line).

With a lot of rounds, it is more difficult to brute forcing the password. But note also that more CPU resources will be needed to authenticate users.

If not specified, the libc will choose the default number of rounds (5000).

The values must be inside the 1000–999,999,999 range.

If only one of the **SHA_CRYPT_MIN_ROUNDS** or **SHA_CRYPT_MAX_ROUNDS** values is set, then this value will be used.

If **SHA_CRYPT_MIN_ROUNDS > SHA_CRYPT_MAX_ROUNDS**, the highest value will be used.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

FILES

- /etc/group
Group account information.
- /etc/gshadow
Secure group account information.
- /etc/login.defs
Shadow password suite configuration.

SEE ALSO

- gpasswd(1), groupadd(8), login.defs(5).**

NAME

chmod – change file mode bits

SYNOPSIS

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

DESCRIPTION

This manual page documents the GNU version of **chmod**. **chmod** changes the file mode bits of each given file according to *mode*, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

The format of a symbolic mode is [**ugo**a...][[-+]=][*perms*...]. . .], where *perms* is either zero or more letters from the set **rwxXst**, or a single letter from the set **ugo**. Multiple symbolic modes can be given, separated by commas.

A combination of the letters **ugo**a controls which users' access to the file will be changed: the user who owns it (**u**), other users in the file's group (**g**), other users not in the file's group (**o**), or all users (**a**). If none of these are given, the effect is as if (**a**) were given, but bits that are set in the umask are not affected.

The operator + causes the selected file mode bits to be added to the existing file mode bits of each file; - causes them to be removed; and = causes them to be added and causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.

The letters **rwxXst** select file mode bits for the affected users: read (**r**), write (**w**), execute (or search for directories) (**x**), execute/search only if the file is a directory or already has execute permission for some user (**X**), set user or group ID on execution (**s**), restricted deletion flag or sticky bit (**t**). Instead of one or more of these letters, you can specify exactly one of the letters **ugo**: the permissions granted to the user who owns the file (**u**), the permissions granted to other users who are members of the file's group (**g**), and the permissions granted to users that are in neither of the two preceding categories (**o**).

A numeric mode is from one to four octal digits (0–7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. The first digit selects the set user ID (4) and set group ID (2) and restricted deletion or sticky (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

chmod never changes the permissions of symbolic links; the **chmod** system call cannot change their permissions. This is not a problem since the permissions of symbolic links are never used. However, for each symbolic link listed on the command line, **chmod** changes the permissions of the pointed-to file. In contrast, **chmod** ignores symbolic links encountered during recursive directory traversals.

SETUID AND SETGID BITS

chmod clears the set-group-ID bit of a regular file if the file's group ID does not match the user's effective group ID or one of the user's supplementary group IDs, unless the user has appropriate privileges. Additional restrictions may cause the set-user-ID and set-group-ID bits of *MODE* or *RFILE* to be ignored. This behavior depends on the policy and functionality of the underlying **chmod** system call. When in doubt, check the underlying system behavior.

For directories **chmod** preserves set-user-ID and set-group-ID bits unless you explicitly specify otherwise. You can set or clear the bits with symbolic modes like **u+s** and **g-s**. To clear these bits for directories with a numeric mode requires an additional leading zero, or leading = like **00755** , or **=755**

RESTRICTED DELETION FLAG OR STICKY BIT

The restricted deletion flag or sticky bit is a single bit, whose interpretation depends on the file type. For directories, it prevents unprivileged users from removing or renaming a file in the directory unless they own the file or the directory; this is called the *restricted deletion flag* for the directory, and is commonly found on world-writable directories like **/tmp**. For regular files on some older systems, the bit saves the program's text image on the swap device so it will load more quickly when run; this is called the *sticky bit*.

OPTIONS

Change the mode of each FILE to MODE. With **--reference**, change the mode of each FILE to that of RFILE.

-c, --changes

like verbose but report only when a change is made

-f, --silent, --quiet

suppress most error messages

-v, --verbose

output a diagnostic for every file processed

--no-preserve-root

do not treat '/' specially (the default)

--preserve-root

fail to operate recursively on '/'

--reference=RFILE

use RFILE's mode instead of MODE values

-R, --recursive

change files and directories recursively

--help display this help and exit

--version

output version information and exit

Each MODE is of the form '[ugoa]*([-=][[rwxXst]*|[ugo]]))|[-=][0-7]+'.

AUTHOR

Written by David MacKenzie and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

chmod(2)

Full documentation <<https://www.gnu.org/software/coreutils/chmod>>
or available locally via: info '(coreutils) chmod invocation'

NAME

chmod, fchmod, fchmodat – change permissions of a file

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/stat.h>
int chmod(const char *pathname, mode_t mode);
int fchmod(int fd, mode_t mode);

#include <fcntl.h>      /* Definition of AT_* constants */
#include <sys/stat.h>

int fchmodat(int dirfd, const char *pathname, mode_t mode, int fla gs);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

fchmod():

Since glibc 2.24:

`_POSIX_C_SOURCE >= 199309L`

glibc 2.19 to glibc 2.23

`_POSIX_C_SOURCE`

glibc 2.16 to glibc 2.19:

`_BSD_SOURCE || _POSIX_C_SOURCE`

glibc 2.12 to glibc 2.16:

`_BSD_SOURCE || _XOPEN_SOURCE >= 500`

`|| _POSIX_C_SOURCE >= 200809L`

glibc 2.11 and earlier:

`_BSD_SOURCE || _XOPEN_SOURCE >= 500`

fchmodat():

Since glibc 2.10:

`_POSIX_C_SOURCE >= 200809L`

Before glibc 2.10:

`_ATFILE_SOURCE`

DESCRIPTION

The **chmod()** and **fchmod()** system calls change a file's mode bits. (The file mode consists of the file permission bits plus the set-user-ID, set-group-ID, and sticky bits.) These system calls differ only in how the file is specified:

- **chmod()** changes the mode of the file specified whose pathname is given in *pathname*, which is dereferenced if it is a symbolic link.
- **fchmod()** changes the mode of the file referred to by the open file descriptor *fd*.

The new file mode is specified in *mode*, which is a bit mask created by ORing together zero or more of the following:

S_ISUID (04000)	set-user-ID (set process effective user ID on execve(2))
S_ISGID (02000)	set-group-ID (set process effective group ID on execve(2) ; mandatory locking, as described in fcntl(2) ; take a new file's group from parent directory, as described in chown(2) and mkdir(2))
S_ISVTX (01000)	sticky bit (restricted deletion flag, as described in unlink(2))
S_IRUSR (00400)	read by owner
S_IWUSR (00200)	write by owner
S_IXUSR (00100)	execute/search by owner ("search" applies for directories, and means that entries within the directory can be accessed)

- S_IRGRP** (00040) read by group
- S_IWGRP** (00020) write by group
- S_IXGRP** (00010) execute/search by group
- S_IROTH** (00004) read by others
- S_IWOTH** (00002) write by others
- S_IXOTH** (00001) execute/search by others

The effective UID of the calling process must match the owner of the file, or the process must be privileged (Linux: it must have the **CAP_FOWNER** capability).

If the calling process is not privileged (Linux: does not have the **CAP_FSETID** capability), and the group of the file does not match the effective group ID of the process or one of its supplementary group IDs, the **S_ISGID** bit will be turned off, but this will not cause an error to be returned.

As a security measure, depending on the filesystem, the set-user-ID and set-group-ID execution bits may be turned off if a file is written. (On Linux, this occurs if the writing process does not have the **CAP_FSETID** capability.) On some filesystems, only the superuser can set the sticky bit, which may have a special meaning. For the sticky bit, and for set-user-ID and set-group-ID bits on directories, see **inode**(7).

On NFS filesystems, restricting the permissions will immediately influence already open files, because the access control is done on the server, but open files are maintained by the client. Widening the permissions may be delayed for other clients if attribute caching is enabled on them.

fchmodat()

The **fchmodat()** system call operates in exactly the same way as **chmod()**, except for the differences described here.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **chmod()** for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **chmod()**).

If *pathname* is absolute, then *dirfd* is ignored.

flags can either be 0, or include the following flag:

AT_SYMLINK_NOFOLLOW

If *pathname* is a symbolic link, do not dereference it: instead operate on the link itself. This flag is not currently implemented.

See **openat**(2) for an explanation of the need for **fchmodat()**.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

Depending on the filesystem, errors other than those listed below can be returned.

The more general errors for **chmod()** are listed below:

EACCES

Search permission is denied on a component of the path prefix. (See also **path_resolution**(7).)

EBADF

(**fchmod()**) The file descriptor *fd* is not valid.

EBADF

(**fchmodat()**) *pathname* is relative but *dirfd* is neither **AT_FDCWD** nor a valid file descriptor.

EFAULT

pathname points outside your accessible address space.

EINVAL

(**fchmodat()**) Invalid flag specified in *flags*.

EIO An I/O error occurred.**ELOOP**

Too many symbolic links were encountered in resolving *pathname*.

ENAMETOOLONG

pathname is too long.

ENOENT

The file does not exist.

ENOMEM

Insufficient kernel memory was available.

ENOTDIR

A component of the path prefix is not a directory.

ENOTDIR

(**fchmodat()**) *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

ENOTSUP

(**fchmodat()**) *flags* specified **AT_SYMLINK_NOFOLLOW**, which is not supported.

EPERM

The effective UID does not match the owner of the file, and the process is not privileged (Linux: it does not have the **CAP_FOWNER** capability).

EPERM

The file is marked immutable or append-only. (See **ioctl_iflags(2)**.)

EROFS

The named file resides on a read-only filesystem.

VERSIONS

fchmodat() was added in Linux 2.6.16; library support was added in glibc 2.4.

STANDARDS

chmod(), **fchmod()**: 4.4BSD, SVr4, POSIX.1-2001i, POSIX.1-2008.

fchmodat(): POSIX.1-2008.

NOTES**C library/kernel differences**

The GNU C library **fchmodat()** wrapper function implements the POSIX-specified interface described in this page. This interface differs from the underlying Linux system call, which does *not* have a *flags* argument.

glibc notes

On older kernels where **fchmodat()** is unavailable, the glibc wrapper function falls back to the use of **chmod()**. When *pathname* is a relative pathname, glibc constructs a pathname based on the symbolic link in */proc/self/fd* that corresponds to the *dirfd* argument.

SEE ALSO

chmod(1), **chown(2)**, **execve(2)**, **open(2)**, **stat(2)**, **inode(7)**, **path_resolution(7)**, **symlink(7)**

NAME

chown – change file owner and group

SYNOPSIS

```
chown [OPTION]... [OWNER][:GROUP] FILE...
chown [OPTION]... --reference=RFILE FILE...
```

DESCRIPTION

This manual page documents the GNU version of **chown**. **chown** changes the user and/or group ownership of each given file. If only an owner (a user name or numeric user ID) is given, that user is made the owner of each given file, and the files' group is not changed. If the owner is followed by a colon and a group name (or numeric group ID), with no spaces between them, the group ownership of the files is changed as well. If a colon but no group name follows the user name, that user is made the owner of the files and the group of the files is changed to that user's login group. If the colon and group are given, but the owner is omitted, only the group of the files is changed; in this case, **chown** performs the same function as **chgrp**. If only a colon is given, or if the entire operand is empty, neither the owner nor the group is changed.

OPTIONS

Change the owner and/or group of each FILE to OWNER and/or GROUP. With **--reference**, change the owner and group of each FILE to those of RFILE.

-c, --changes

like verbose but report only when a change is made

-f, --silent, --quiet

suppress most error messages

-v, --verbose

output a diagnostic for every file processed

--dereference

affect the referent of each symbolic link (this is the default), rather than the symbolic link itself

-h, --no-dereference

affect symbolic links instead of any referenced file (useful only on systems that can change the ownership of a symlink)

--from=CURRENT_OWNER:CURRENT_GROUP

change the owner and/or group of each file only if its current owner and/or group match those specified here. Either may be omitted, in which case a match is not required for the omitted attribute

--no-preserve-root

do not treat '/' specially (the default)

--preserve-root

fail to operate recursively on '/'

--reference=RFILE

use RFILE's owner and group rather than specifying OWNER:GROUP values

-R, --recursive

operate on files and directories recursively

The following options modify how a hierarchy is traversed when the **-R** option is also specified. If more than one is specified, only the final one takes effect.

-H if a command line argument is a symbolic link to a directory, traverse it

-L traverse every symbolic link to a directory encountered

-P do not traverse any symbolic links (default)

--help display this help and exit

--version

output version information and exit

Owner is unchanged if missing. Group is unchanged if missing, but changed to login group if implied by a ':' following a symbolic OWNER. OWNER and GROUP may be numeric as well as symbolic.

EXAMPLES

chown root /u

Change the owner of /u to "root".

chown root:staff /u

Likewise, but also change its group to "staff".

chown -hR root /u

Change the owner of /u and subfiles to "root".

AUTHOR

Written by David MacKenzie and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

chown(2)

Full documentation <<https://www.gnu.org/software/coreutils/chown>>
or available locally via: info '(coreutils) chown invocation'

NAME

chown, fchown, lchown, fchownat – change ownership of a file

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int chown(const char *pathname, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *pathname, uid_t owner, gid_t group);

#include <fcntl.h>      /* Definition of AT_* constants */
#include <unistd.h>

int fchownat(int dirfd, const char *pathname,
             uid_t owner, gid_t group, int flags);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
fchown(), lchown():
/* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L
|| _XOPEN_SOURCE >= 500
|| /* glibc <= 2.19: */ _BSD_SOURCE

fchownat():
Since glibc 2.10:
_POSIX_C_SOURCE >= 200809L
Before glibc 2.10:
_ATFILE_SOURCE
```

DESCRIPTION

These system calls change the owner and group of a file. The **chown()**, **fchown()**, and **lchown()** system calls differ only in how the file is specified:

- **chown()** changes the ownership of the file specified by *pathname*, which is dereferenced if it is a symbolic link.
- **fchown()** changes the ownership of the file referred to by the open file descriptor *fd*.
- **lchown()** is like **chown()**, but does not dereference symbolic links.

Only a privileged process (Linux: one with the **CAP_CHOWN** capability) may change the owner of a file. The owner of a file may change the group of the file to any group of which that owner is a member. A privileged process (Linux: with **CAP_CHOWN**) may change the group arbitrarily.

If the *owner* or *group* is specified as -1 , then that ID is not changed.

When the owner or group of an executable file is changed by an unprivileged user, the **S_ISUID** and **S_ISGID** mode bits are cleared. POSIX does not specify whether this also should happen when root does the **chown()**; the Linux behavior depends on the kernel version, and since Linux 2.2.13, root is treated like other users. In case of a non-group-executable file (i.e., one for which the **S_IXGRP** bit is not set) the **S_ISGID** bit indicates mandatory locking, and is not cleared by a **chown()**.

When the owner or group of an executable file is changed (by any user), all capability sets for the file are cleared.

fchownat()

The **fchownat()** system call operates in exactly the same way as **chown()**, except for the differences described here.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **chown()** for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **chown()**).

If *pathname* is absolute, then *dirfd* is ignored.

The *flags* argument is a bit mask created by ORing together 0 or more of the following values;

AT_EMPTY_PATH (since Linux 2.6.39)

If *pathname* is an empty string, operate on the file referred to by *dirfd* (which may have been obtained using the **open(2)** **O_PATH** flag). In this case, *dirfd* can refer to any type of file, not just a directory. If *dirfd* is **AT_FDCWD**, the call operates on the current working directory. This flag is Linux-specific; define **_GNU_SOURCE** to obtain its definition.

AT_SYMLINK_NOFOLLOW

If *pathname* is a symbolic link, do not dereference it: instead operate on the link itself, like **lchown()**. (By default, **fchownat()** dereferences symbolic links, like **chown()**.)

See **openat(2)** for an explanation of the need for **fchownat()**.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

Depending on the filesystem, errors other than those listed below can be returned.

The more general errors for **chown()** are listed below.

EACCES

Search permission is denied on a component of the path prefix. (See also **path_resolution(7)**.)

EBADF

(**fchown()**) *fd* is not a valid open file descriptor.

EBADF

(**fchownat()**) *pathname* is relative but *dirfd* is neither **AT_FDCWD** nor a valid file descriptor.

EFAULT

pathname points outside your accessible address space.

EINVAL

(**fchownat()**) Invalid flag specified in *flags*.

EIO

(**fchown()**) A low-level I/O error occurred while modifying the inode.

ELOOP

Too many symbolic links were encountered in resolving *pathname*.

ENAMETOOLONG

pathname is too long.

ENOENT

The file does not exist.

ENOMEM

Insufficient kernel memory was available.

ENOTDIR

A component of the path prefix is not a directory.

ENOTDIR

(**fchownat()**) *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

EPERM

The calling process did not have the required permissions (see above) to change owner and/or group.

EPERM

The file is marked immutable or append-only. (See [ioctl_iflags\(2\)](#).)

EROFS

The named file resides on a read-only filesystem.

VERSIONS

fchownat() was added in Linux 2.6.16; library support was added in glibc 2.4.

STANDARDS

chown(), **fchown()**, **lchown()**: 4.4BSD, SVr4, POSIX.1-2001, POSIX.1-2008.

The 4.4BSD version can be used only by the superuser (that is, ordinary users cannot give away files).

fchownat(): POSIX.1-2008.

NOTES**Ownership of new files**

When a new file is created (by, for example, **open(2)** or **mkdir(2)**), its owner is made the same as the filesystem user ID of the creating process. The group of the file depends on a range of factors, including the type of filesystem, the options used to mount the filesystem, and whether or not the set-group-ID mode bit is enabled on the parent directory. If the filesystem supports the **-o gr pid** (or, synonymously **-o bsd-groups**) and **-o nogrpid** (or, synonymously **-o sysvgroups**) **mount(8)** options, then the rules are as follows:

- If the filesystem is mounted with **-o gr pid**, then the group of a new file is made the same as that of the parent directory.
- If the filesystem is mounted with **-o nogrpid** and the set-group-ID bit is disabled on the parent directory, then the group of a new file is made the same as the process's filesystem GID.
- If the filesystem is mounted with **-o nogrpid** and the set-group-ID bit is enabled on the parent directory, then the group of a new file is made the same as that of the parent directory.

As at Linux 4.12, the **-o gr pid** and **-o nogrpid** mount options are supported by ext2, ext3, ext4, and XFS. Filesystems that don't support these mount options follow the **-o nogrpid** rules.

glibc notes

On older kernels where **fchownat()** is unavailable, the glibc wrapper function falls back to the use of **chown()** and **lchown()**. When *pathname* is a relative pathname, glibc constructs a pathname based on the symbolic link in */proc/self/fd* that corresponds to the *dirfd* argument.

NFS

The **chown()** semantics are deliberately violated on NFS filesystems which have UID mapping enabled. Additionally, the semantics of all system calls which access the file contents are violated, because **chown()** may cause immediate access revocation on already open files. Client side caching may lead to a delay between the time where ownership have been changed to allow access for a user and the time where the file can actually be accessed by the user on other clients.

Historical details

The original Linux **chown()**, **fchown()**, and **lchown()** system calls supported only 16-bit user and group IDs. Subsequently, Linux 2.4 added **chown32()**, **fchown32()**, and **lchown32()**, supporting 32-bit IDs. The glibc **chown()**, **fchown()**, and **lchown()** wrapper functions transparently deal with the variations across kernel versions.

Before Linux 2.1.81 (except 2.1.46), **chown()** did not follow symbolic links. Since Linux 2.1.81, **chown()** does follow symbolic links, and there is a new system call **lchown()** that does not follow symbolic links. Since Linux 2.1.86, this new call (that has the same semantics as the old **chown()**) has got the same syscall number, and **chown()** got the newly introduced number.

EXAMPLES

The following program changes the ownership of the file named in its second command-line argument to the value specified in its first command-line argument. The new owner can be specified either as a numeric

user ID, or as a username (which is converted to a user ID by using **getpwnam(3)** to perform a lookup in the system password file).

Program source

```
#include <pwd.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
    char             *endptr;
    uid_t            uid;
    struct passwd   *pwd;

    if (argc != 3 || argv[1][0] == '\0') {
        fprintf(stderr, "%s <owner> <file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    uid = strtol(argv[1], &endptr, 10); /* Allow a numeric string */

    if (*endptr != '\0') {           /* Was not pure numeric string */
        pwd = getpwnam(argv[1]);    /* Try getting UID for username */
        if (pwd == NULL) {
            perror("getpwnam");
            exit(EXIT_FAILURE);
        }

        uid = pwd->pw_uid;
    }

    if (chown(argv[2], uid, -1) == -1) {
        perror("chown");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

SEE ALSO

chgrp(1), **chown(1)**, **chmod(2)**, **flock(2)**, **path_resolution(7)**, **symlink(7)**

NAME

chown, fchown, lchown, fchownat – change ownership of a file

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int chown(const char *pathname, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
int lchown(const char *pathname, uid_t owner, gid_t group);

#include <fcntl.h>      /* Definition of AT_* constants */
#include <unistd.h>

int fchownat(int dirfd, const char *pathname,
             uid_t owner, gid_t group, int flags);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
fchown(), lchown():
/* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L
|| _XOPEN_SOURCE >= 500
|| /* glibc <= 2.19: */ _BSD_SOURCE

fchownat():
Since glibc 2.10:
_POSIX_C_SOURCE >= 200809L
Before glibc 2.10:
_ATFILE_SOURCE
```

DESCRIPTION

These system calls change the owner and group of a file. The **chown()**, **fchown()**, and **lchown()** system calls differ only in how the file is specified:

- **chown()** changes the ownership of the file specified by *pathname*, which is dereferenced if it is a symbolic link.
- **fchown()** changes the ownership of the file referred to by the open file descriptor *fd*.
- **lchown()** is like **chown()**, but does not dereference symbolic links.

Only a privileged process (Linux: one with the **CAP_CHOWN** capability) may change the owner of a file. The owner of a file may change the group of the file to any group of which that owner is a member. A privileged process (Linux: with **CAP_CHOWN**) may change the group arbitrarily.

If the *owner* or *group* is specified as -1 , then that ID is not changed.

When the owner or group of an executable file is changed by an unprivileged user, the **S_ISUID** and **S_ISGID** mode bits are cleared. POSIX does not specify whether this also should happen when root does the **chown()**; the Linux behavior depends on the kernel version, and since Linux 2.2.13, root is treated like other users. In case of a non-group-executable file (i.e., one for which the **S_IXGRP** bit is not set) the **S_ISGID** bit indicates mandatory locking, and is not cleared by a **chown()**.

When the owner or group of an executable file is changed (by any user), all capability sets for the file are cleared.

fchownat()

The **fchownat()** system call operates in exactly the same way as **chown()**, except for the differences described here.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **chown()** for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **chown()**).

If *pathname* is absolute, then *dirfd* is ignored.

The *flags* argument is a bit mask created by ORing together 0 or more of the following values;

AT_EMPTY_PATH (since Linux 2.6.39)

If *pathname* is an empty string, operate on the file referred to by *dirfd* (which may have been obtained using the **open(2)** **O_PATH** flag). In this case, *dirfd* can refer to any type of file, not just a directory. If *dirfd* is **AT_FDCWD**, the call operates on the current working directory. This flag is Linux-specific; define **_GNU_SOURCE** to obtain its definition.

AT_SYMLINK_NOFOLLOW

If *pathname* is a symbolic link, do not dereference it: instead operate on the link itself, like **lchown()**. (By default, **fchownat()** dereferences symbolic links, like **chown()**.)

See **openat(2)** for an explanation of the need for **fchownat()**.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

Depending on the filesystem, errors other than those listed below can be returned.

The more general errors for **chown()** are listed below.

EACCES

Search permission is denied on a component of the path prefix. (See also **path_resolution(7)**.)

EBADF

(**fchown()**) *fd* is not a valid open file descriptor.

EBADF

(**fchownat()**) *pathname* is relative but *dirfd* is neither **AT_FDCWD** nor a valid file descriptor.

EFAULT

pathname points outside your accessible address space.

EINVAL

(**fchownat()**) Invalid flag specified in *flags*.

EIO

(**fchown()**) A low-level I/O error occurred while modifying the inode.

ELOOP

Too many symbolic links were encountered in resolving *pathname*.

ENAMETOOLONG

pathname is too long.

ENOENT

The file does not exist.

ENOMEM

Insufficient kernel memory was available.

ENOTDIR

A component of the path prefix is not a directory.

ENOTDIR

(**fchownat()**) *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

EPERM

The calling process did not have the required permissions (see above) to change owner and/or group.

EPERM

The file is marked immutable or append-only. (See [ioctl_iflags\(2\)](#).)

EROFS

The named file resides on a read-only filesystem.

VERSIONS

fchownat() was added in Linux 2.6.16; library support was added in glibc 2.4.

STANDARDS

chown(), **fchown()**, **lchown()**: 4.4BSD, SVr4, POSIX.1-2001, POSIX.1-2008.

The 4.4BSD version can be used only by the superuser (that is, ordinary users cannot give away files).

fchownat(): POSIX.1-2008.

NOTES**Ownership of new files**

When a new file is created (by, for example, **open(2)** or **mkdir(2)**), its owner is made the same as the filesystem user ID of the creating process. The group of the file depends on a range of factors, including the type of filesystem, the options used to mount the filesystem, and whether or not the set-group-ID mode bit is enabled on the parent directory. If the filesystem supports the **-o gr pid** (or, synonymously **-o bsd-groups**) and **-o nogrpid** (or, synonymously **-o sysvgroups**) **mount(8)** options, then the rules are as follows:

- If the filesystem is mounted with **-o gr pid**, then the group of a new file is made the same as that of the parent directory.
- If the filesystem is mounted with **-o nogrpid** and the set-group-ID bit is disabled on the parent directory, then the group of a new file is made the same as the process's filesystem GID.
- If the filesystem is mounted with **-o nogrpid** and the set-group-ID bit is enabled on the parent directory, then the group of a new file is made the same as that of the parent directory.

As at Linux 4.12, the **-o gr pid** and **-o nogrpid** mount options are supported by ext2, ext3, ext4, and XFS. Filesystems that don't support these mount options follow the **-o nogrpid** rules.

glibc notes

On older kernels where **fchownat()** is unavailable, the glibc wrapper function falls back to the use of **chown()** and **lchown()**. When *pathname* is a relative pathname, glibc constructs a pathname based on the symbolic link in */proc/self/fd* that corresponds to the *dirfd* argument.

NFS

The **chown()** semantics are deliberately violated on NFS filesystems which have UID mapping enabled. Additionally, the semantics of all system calls which access the file contents are violated, because **chown()** may cause immediate access revocation on already open files. Client side caching may lead to a delay between the time where ownership have been changed to allow access for a user and the time where the file can actually be accessed by the user on other clients.

Historical details

The original Linux **chown()**, **fchown()**, and **lchown()** system calls supported only 16-bit user and group IDs. Subsequently, Linux 2.4 added **chown32()**, **fchown32()**, and **lchown32()**, supporting 32-bit IDs. The glibc **chown()**, **fchown()**, and **lchown()** wrapper functions transparently deal with the variations across kernel versions.

Before Linux 2.1.81 (except 2.1.46), **chown()** did not follow symbolic links. Since Linux 2.1.81, **chown()** does follow symbolic links, and there is a new system call **lchown()** that does not follow symbolic links. Since Linux 2.1.86, this new call (that has the same semantics as the old **chown()**) has got the same syscall number, and **chown()** got the newly introduced number.

EXAMPLES

The following program changes the ownership of the file named in its second command-line argument to the value specified in its first command-line argument. The new owner can be specified either as a numeric

user ID, or as a username (which is converted to a user ID by using **getpwnam(3)** to perform a lookup in the system password file).

Program source

```
#include <pwd.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
    char             *endptr;
    uid_t            uid;
    struct passwd   *pwd;

    if (argc != 3 || argv[1][0] == '\0') {
        fprintf(stderr, "%s <owner> <file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    uid = strtol(argv[1], &endptr, 10); /* Allow a numeric string */

    if (*endptr != '\0') {           /* Was not pure numeric string */
        pwd = getpwnam(argv[1]);    /* Try getting UID for username */
        if (pwd == NULL) {
            perror("getpwnam");
            exit(EXIT_FAILURE);
        }

        uid = pwd->pw_uid;
    }

    if (chown(argv[2], uid, -1) == -1) {
        perror("chown");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

SEE ALSO

chgrp(1), **chown(1)**, **chmod(2)**, **flock(2)**, **path_resolution(7)**, **symlink(7)**

NAME

chpasswd – update passwords in batch mode

SYNOPSIS

chpasswd [*options*]

DESCRIPTION

The **chpasswd** command reads a list of user name and password pairs from standard input and uses this information to update a group of existing users. Each line is of the format:

user_name:password

By default the passwords must be supplied in clear-text, and are encrypted by **chpasswd**. Also the password age will be updated, if present.

By default, passwords are encrypted by PAM, but (even if not recommended) you can select a different encryption method with the **-e**, **-m**, or **-c** options.

Except when PAM is used to encrypt the passwords, **chpasswd** first updates all the passwords in memory, and then commits all the changes to disk if no errors occurred for any user.

When PAM is used to encrypt the passwords (and update the passwords in the system database) then if a password cannot be updated **chpasswd** continues updating the passwords of the next users, and will return an error code on exit.

This command is intended to be used in a large system environment where many accounts are created at a single time.

OPTIONS

The options which apply to the **chpasswd** command are:

-c, --crypt-method *METHOD*

Use the specified method to encrypt the passwords.

The available methods are DES, MD5, NONE, and SHA256 or SHA512 if your libc support these methods.

By default, PAM is used to encrypt the passwords.

-e, --encrypted

Supplied passwords are in encrypted form.

-h, --help

Display help message and exit.

-m, --md5

Use MD5 encryption instead of DES when the supplied passwords are not encrypted.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-s, --sha-rounds *ROUNDS*

Use the specified number of rounds to encrypt the passwords.

The value 0 means that the system will choose the default number of rounds for the crypt method (5000).

A minimal value of 1000 and a maximal value of 999,999,999 will be enforced.

You can only use this option with the SHA256 or SHA512 crypt method.

By default, the number of rounds is defined by the **SHA_CRYPT_MIN_ROUNDS** and

SHA_CRYPT_MAX_ROUNDS variables in /etc/login.defs.

CAVEATS

Remember to set permissions or umask to prevent readability of unencrypted files by other users.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

SHA_CRYPT_MIN_ROUNDS (number), **SHA_CRYPT_MAX_ROUNDS** (number)

When **ENCRYPT_METHOD** is set to *SHA256* or *SHA512*, this defines the number of SHA rounds used by the encryption algorithm by default (when the number of rounds is not specified on the command line).

With a lot of rounds, it is more difficult to brute forcing the password. But note also that more CPU resources will be needed to authenticate users.

If not specified, the libc will choose the default number of rounds (5000).

The values must be inside the 1000–999,999,999 range.

If only one of the **SHA_CRYPT_MIN_ROUNDS** or **SHA_CRYPT_MAX_ROUNDS** values is set, then this value will be used.

If **SHA_CRYPT_MIN_ROUNDS > SHA_CRYPT_MAX_ROUNDS**, the highest value will be used.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

FILES

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

/etc/login.defs

Shadow password suite configuration.

/etc/pam.d/chpasswd

PAM configuration for **chpasswd**.

SEE ALSO

passwd(1), **newusers(8)**, **login.defs(5)**, **useradd(8)**.

NAME

chroot – change root directory

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int chroot(const char *path);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
chroot():
  Since glibc 2.2.2:
    _XOPEN_SOURCE && !(_POSIX_C_SOURCE >= 200112L)
    || /* Since glibc 2.20: */ _DEFAULT_SOURCE
    || /* glibc <= 2.19: */ _BSD_SOURCE
  Before glibc 2.2.2:
    none
```

DESCRIPTION

chroot() changes the root directory of the calling process to that specified in *path*. This directory will be used for pathnames beginning with */*. The root directory is inherited by all children of the calling process.

Only a privileged process (Linux: one with the **CAP_SYS_CHROOT** capability in its user namespace) may call **chroot()**.

This call changes an ingredient in the pathname resolution process and does nothing else. In particular, it is not intended to be used for any kind of security purpose, neither to fully sandbox a process nor to restrict filesystem system calls. In the past, **chroot()** has been used by daemons to restrict themselves prior to passing paths supplied by untrusted users to system calls such as **open(2)**. However, if a folder is moved out of the chroot directory, an attacker can exploit that to get out of the chroot directory as well. The easiest way to do that is to **chdir(2)** to the to-be-moved directory, wait for it to be moved out, then open a path like *../../../../etc/passwd*.

A slightly trickier variation also works under some circumstances if **chdir(2)** is not permitted. If a daemon allows a "chroot directory" to be specified, that usually means that if you want to prevent remote users from accessing files outside the chroot directory, you must ensure that folders are never moved out of it.

This call does not change the current working directory, so that after the call *'.'* can be outside the tree rooted at *'/'*. In particular, the superuser can escape from a "chroot jail" by doing:

```
mkdir foo; chroot foo; cd ..
```

This call does not close open file descriptors, and such file descriptors may allow access to files outside the chroot tree.

RETURN VALUE

On success, zero is returned. On error, *-1* is returned, and *errno* is set to indicate the error.

ERRORS

Depending on the filesystem, other errors can be returned. The more general errors are listed below:

EACCES

Search permission is denied on a component of the path prefix. (See also **path_resolution(7)**.)

EFAULT

path points outside your accessible address space.

EIO

An I/O error occurred.

ELOOP

Too many symbolic links were encountered in resolving *path*.

ENAMETOOLONG

path is too long.

ENOENT

The file does not exist.

ENOMEM

Insufficient kernel memory was available.

ENOTDIR

A component of *path* is not a directory.

EPERM

The caller has insufficient privilege.

STANDARDS

SVr4, 4.4BSD, SUSv2 (marked LEGACY). This function is not part of POSIX.1-2001.

NOTES

A child process created via **fork(2)** inherits its parent's root directory. The root directory is left unchanged by **execve(2)**.

The magic symbolic link, */proc/[pid]/root*, can be used to discover a process's root directory; see **proc(5)** for details.

FreeBSD has a stronger **jail()** system call.

SEE ALSO

chroot(1), **chdir(2)**, **pivot_root(2)**, **path_resolution(7)**, **switch_root(8)**

NAME

chroot – run command or interactive shell with special root directory

SYNOPSIS

chroot [*OPTION*] *NEWROOT* [*COMMAND* [*ARG*]...]
chroot *OPTION*

DESCRIPTION

Run *COMMAND* with root directory set to *NEWROOT*.

--groups=*G_LIST*
specify supplementary groups as g1,g2,...,gN
--userspec=*USER:GROUP*
specify user and group (ID or name) to use
--skip-chdir
do not change working directory to '/'
--help display this help and exit
--version
output version information and exit

If no command is given, run '\$SHELL' -i (default: '/bin/sh -i').

AUTHOR

Written by Roland McGrath.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>. This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

chroot(2)

Full documentation <<https://www.gnu.org/software/coreutils/chroot>>
or available locally via: info '(coreutils) chroot invocation'

NAME

cron – daemon to execute scheduled commands (Vixie Cron)

SYNOPSIS

cron [**-f**] [**-l**] [**-L loglevel**]

DESCRIPTION

cron is started automatically from /etc/init.d on entering multi-user runlevels.

OPTIONS

- f** Stay in foreground mode, don't daemonize.
- P** Don't set PATH for child processes. Let it inherit instead.
- l** Enable LSB compliant names for /etc/cron.d files. This setting, however, does not affect the parsing of files under /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly or /etc/cron.monthly.
- n** Include the FQDN in the subject when sending mails. By default, cron will abbreviate the host-name.

-L loglevel

Tell cron what to log about **jobs** (errors are logged regardless of this value) as the sum of the following values:

- 1** will log the start of all cron jobs
- 2** will log the end of all cron jobs
- 4** will log all failed jobs (exit status != 0)
- 8** will log the process number of all cron jobs

The default is to log the start of all jobs (1). Logging will be disabled if *levels* is set to zero (0). A value of fifteen (15) will select all options.

NOTES

cron searches its spool area (/var/spool/cron/crontabs) for crontab files (which are named after accounts in /etc/passwd); crontabs found are loaded into memory. Note that crontabs in this directory should not be accessed directly - the *crontab* command should be used to access and update them.

cron also reads /etc/crontab, which is in a slightly different format (see *crontab(5)*). In Debian, the content of /etc/crontab is predefined to run programs under /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly. This configuration is specific to Debian, see the note under **DEBIAN SPECIFIC** below.

Additionally, in Debian, *cron* reads the files in the /etc/cron.d directory. *cron* treats the files in /etc/cron.d as in the same way as the /etc/crontab file (they follow the special format of that file, i.e. they include the *user* field). However, they are independent of /etc/crontab: they do not, for example, inherit environment variable settings from it. This change is specific to Debian see the note under **DEBIAN SPECIFIC** below.

Like /etc/crontab, the files in the /etc/cron.d directory are monitored for changes. In general, the system administrator should not use /etc/cron.d/, but use the standard system crontab /etc/crontab.

/etc/crontab and the files in /etc/cron.d must be owned by root, and must not be group- or other-writable. In contrast to the spool area, the files under /etc/cron.d or the files under /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly may also be symlinks, provided that both the symlink and the file it points to are owned by root. The files under /etc/cron.d do not need to be executable, while the files under /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly do, as they are run by *run-parts* (see *run-parts(8)* for more information).

cron then wakes up every minute, examining all stored crontabs, checking each command to see if it should be run in the current minute. When executing commands, any output is mailed to the owner of the crontab (or to the user named in the MAILTO environment variable in the crontab, if such exists) from the owner of

the crontab (or from the email address given in the MAILFROM environment variable in the crontab, if such exists). The children copies of cron running these processes have their name coerced to uppercase, as will be seen in the syslog and ps output.

Additionally, *cron* checks each minute to see if its spool directory's modtime (or the modtime on the */etc/crontab* file) has changed, and if it has, *cron* will then examine the modtime on all crontabs and reload those which have changed. Thus *cron* need not be restarted whenever a crontab file is modified. Note that the *crontab(1)* command updates the modtime of the spool directory whenever it changes a crontab.

Special considerations exist when the clock is changed by less than 3 hours, for example at the beginning and end of daylight savings time. If the time has moved forwards, those jobs which would have run in the time that was skipped will be run soon after the change. Conversely, if the time has moved backwards by less than 3 hours, those jobs that fall into the repeated time will not be re-run.

Only jobs that run at a particular time (not specified as @hourly, nor with '*' in the hour or minute specifier) are affected. Jobs which are specified with wildcards are run based on the new time immediately.

Clock changes of more than 3 hours are considered to be corrections to the clock, and the new time is used immediately.

cron logs its action to the syslog facility 'cron', and logging may be controlled using the standard *syslogd(8)* facility.

ENVIRONMENT

If configured in */etc/default/cron* in Debian systems, the *cron* daemon localisation settings environment can be managed through the use of */etc/environment* or through the use of */etc/default/locale* with values from the latter overriding values from the former. These files are read and they will be used to setup the LANG, LC_ALL, and LC_CTYPE environment variables. These variables are then used to set the charset of mails, which defaults to 'C'.

This does **NOT** affect the environment of tasks running under cron. For more information on how to modify the environment of tasks, consult *crontab(5)*.

The daemon will use, if present, the definition from */etc/timezone* for the timezone.

The environment can be redefined in user's crontab definitions but *cron* will only handle tasks in a single timezone.

DEBIAN SPECIFIC

Debian introduces some changes to *cron* that were not originally available upstream. The most significant changes introduced are:

- Support for */etc/cron.{hourly,daily,weekly,monthly}* via */etc/crontab*,
- Support for */etc/cron.d* (drop-in dir for package crontabs),
- PAM support,
- SELinux support,
- auditlog support,
- DST and other time-related changes/fixes,
- SGID *crontab(1)* instead of SUID root,
- Debian-specific file locations and commands,
- Debian-specific configuration (*/etc/default/cron*),
- numerous other smaller features and fixes.

Support for */etc/cron.hourly*, */etc/cron.daily*, */etc/cron.weekly* and */etc/cron.monthly* is provided in Debian through the default setting of the */etc/crontab* file (see the system-wide example in *crontab(5)*). The default

system-wide crontab contains four tasks: run every hour, every day, every week and every month. Each of these tasks will execute **run-parts** providing each one of the directories as an argument. These tasks are disabled if **anacron** is installed (except for the hourly task) to prevent conflicts between both daemons.

As described above, the files under these directories have to pass some sanity checks including the following: be executable, be owned by root, not be writable by group or other and, if symlinks, point to files owned by root. Additionally, the file names must conform to the filename requirements of **run-parts**: they must be entirely made up of letters, digits and can only contain the special signs underscores ('_') and hyphens ('-'). Any file that does not conform to these requirements will not be executed by **run-parts**. For example, any file containing dots will be ignored. This is done to prevent cron from running any of the files that are left by the Debian package management system when handling files in /etc/cron.d/ as configuration files (i.e. files ending in .dpkg-dist, .dpkg-orig, .dpkg-old, and .dpkg-new).

This feature can be used by system administrators and packages to include tasks that will be run at defined intervals. Files created by packages in these directories should be named after the package that supplies them.

Support for /etc/cron.d is included in the *cron* daemon itself, which handles this location as the system-wide crontab spool. This directory can contain any file defining tasks following the format used in /etc/crontab, i.e. unlike the user cron spool, these files must provide the username to run the task as in the task definition.

Files in this directory have to be owned by root, do not need to be executable (they are configuration files, just like /etc/crontab) and must conform to the same naming convention as used by *run-parts*(8) : they must consist solely of upper- and lower-case letters, digits, underscores, and hyphens. This means that they **cannot** contain any dots. If the **-I** option is specified to *cron* (this option can be setup through /etc/default/cron, see below), then they must conform to the LSB namespace specification, exactly as in the **--lsb-sysinit** option in *run-parts*.

The intended purpose of this feature is to allow packages that require finer control of their scheduling than the /etc/cron.{hourly,daily,weekly,monthly} directories to add a crontab file to /etc/cron.d. Such files should be named after the package that supplies them.

Also, the default configuration of *cron* is controlled by /etc/default/cron which is read by the init.d script that launches the *cron* daemon. This file determines whether *cron* will read the system's environment variables and makes it possible to add additional options to the *cron* program before it is executed, either to configure its logging or to define how it will treat the files under /etc/cron.d.

SEE ALSO

crontab(1), crontab(5), run-parts(8)

AUTHOR

Paul Vixie <paul@vix.com> is the author of *cron* and original creator of this manual page. This page has also been modified for Debian by Steve Greenland, Javier Fernandez-Sanguino and Christian Kastner.

NAME

`crontab` – maintain crontab files for individual users (Vixie Cron)

SYNOPSIS

```
crontab [ -u user ] file
crontab [ -u user ] [ -i ] { -e | -l | -r }
```

DESCRIPTION

crontab is the program used to install, deinstall or list the tables used to drive the *cron*(8) daemon in Vixie Cron. Each user can have their own crontab, and though these are files in */var/spool/cron/crontabs*, they are not intended to be edited directly.

If the */etc/cron.allow* file exists, then you must be listed (one user per line) therein in order to be allowed to use this command. If the */etc/cron.allow* file does not exist but the */etc/cron.deny* file does exist, then you must **not** be listed in the */etc/cron.deny* file in order to use this command.

If neither of these files exists, then depending on site-dependent configuration parameters, only the super user will be allowed to use this command, or all users will be able to use this command.

If both files exist then */etc/cron.allow* takes precedence. Which means that */etc/cron.deny* is not considered and your user must be listed in */etc/cron.allow* in order to be able to use the crontab.

Regardless of the existence of any of these files, the root administrative user is always allowed to setup a crontab. For standard Debian systems, all users may use this command.

If the *-u* option is given, it specifies the name of the user whose crontab is to be used (when listing) or modified (when editing). If this option is not given, *crontab* examines "your" crontab, i.e., the crontab of the person executing the command. Note that *su*(8) can confuse *crontab* and that if you are running inside of *su*(8) you should always use the *-u* option for safety's sake.

The first form of this command is used to install a new crontab from some named file or standard input if the pseudo-filename “*-*” is given.

The *-l* option causes the current crontab to be displayed on standard output. See the note under **DEBIAN SPECIFIC** below.

The *-r* option causes the current crontab to be removed.

The *-e* option is used to edit the current crontab using the editor specified by the *VISUAL* or *EDITOR* environment variables. After you exit from the editor, the modified crontab will be installed automatically. If neither of the environment variables is defined, then the default editor */usr/bin/editor* is used.

The *-i* option modifies the *-r* option to prompt the user for a 'y/Y' response before actually removing the crontab.

DEBIAN SPECIFIC

The "out-of-the-box" behaviour for *crontab -l* is to display the three line "DO NOT EDIT THIS FILE" header that is placed at the beginning of the crontab when it is installed. The problem is that it makes the sequence

```
crontab -l | crontab -
```

non-idempotent — you keep adding copies of the header. This causes pain to scripts that use sed to edit a crontab. Therefore, the default behaviour of the *-l* option has been changed to not output such header. You may obtain the original behaviour by setting the environment variable **CRONTAB_NOHEADER** to 'N', which will cause the *crontab -l* command to emit the extraneous header.

SEE ALSO

crontab(5), *cron*(8)

FILES

```
/etc/cron.allow
/etc/cron.deny
/var/spool/cron/crontabs
```

The files */etc/cron.allow* and */etc/cron.deny* if, they exist, must be either world-readable, or readable by group “*crontab*”. If they are not, then cron will deny access to all users until the permissions are fixed.

There is one file for each user’s crontab under the */var/spool/cron/crontabs* directory. Users are not allowed to edit the files under that directory directly to ensure that only users allowed by the system to run periodic tasks can add them, and only syntactically correct crontabs will be written there. This is enforced by having the directory writable only by the *crontab* group and configuring *crontab* command with the setgid bit set for that specific group.

STANDARDS

The *crontab* command conforms to IEEE Std1003.2-1992 (“POSIX”). This new command syntax differs from previous versions of Vixie Cron, as well as from the classic SVR3 syntax.

DIAGNOSTICS

A fairly informative usage message appears if you run it with a bad command line.

cron requires that each entry in a crontab end in a newline character. If the last entry in a crontab is missing the newline, cron will consider the crontab (at least partially) broken and refuse to install it.

The files under */var/spool/cron/crontabs* are named based on the user’s account name. Crontab jobs will not be run for users whose accounts have been renamed either due to changes in the local system or because they are managed through a central user database (external to the system, for example an LDAP directory).

AUTHOR

Paul Vixie <paul@vix.com> is the author of *cron* and original creator of this manual page. This page has also been modified for Debian by Steve Greenland, Javier Fernandez-Sanguino and Christian Kastner.

NAME

`crontab` – tables for driving cron

DESCRIPTION

A `crontab` file contains instructions to the `cron(8)` daemon of the general form: “run this command at this time on this date”. Each user has their own crontab, and commands in any given crontab will be executed as the user who owns the crontab. Uucp and News will usually have their own crontabs, eliminating the need for explicitly running `su(1)` as part of a cron command.

Blank lines and leading spaces and tabs are ignored. Lines whose first non-space character is a hash-sign (#) are comments, and are ignored. Note that comments are not allowed on the same line as cron commands, since they will be taken to be part of the command. Similarly, comments are not allowed on the same line as environment variable settings.

An active line in a crontab will be either an environment setting or a cron command. The crontab file is parsed from top to bottom, so any environment settings will affect only the cron commands below them in the file. An environment setting is of the form,

```
name = value
```

where the spaces around the equal-sign (=) are optional, and any subsequent non-leading spaces in *value* will be part of the value assigned to *name*. The *value* string may be placed in quotes (single or double, but matching) to preserve leading or trailing blanks. To define an empty variable, quotes **must** be used.

The *value* string is **not** parsed for environmental substitutions or replacement of variables or tilde(~) expansion, thus lines like

```
PATH = $HOME/bin:$PATH
PATH = ~/bin:/usr/bin:/bin
```

will not work as you might expect. And neither will this work

```
A=1
B=2
C=$A $B
```

There will not be any substitution for the defined variables in the last value.

Several environment variables are set up automatically by the `cron(8)` daemon. SHELL is set to /bin/sh, and LOGNAME and HOME are set from the /etc/passwd line of the crontab’s owner. PATH is inherited from the environment. HOME, SHELL, and PATH may be overridden by settings in the crontab; LOGNAME is the user that the job is running from, and may not be changed.

(Another note: the LOGNAME variable is sometimes called USER on BSD systems... on these systems, USER will be set also.)

In addition to LOGNAME, HOME, and SHELL, `cron(8)` will look at MAILTO and MAILFROM if it has any reason to send mail as a result of running commands in “this” crontab.

If MAILTO is defined (and non-empty), mail is sent to the user so named. MAILTO may also be used to direct mail to multiple recipients by separating recipient users with a comma. If MAILTO is defined but empty (MAILTO=""), no mail will be sent. Otherwise mail is sent to the owner of the crontab.

If MAILFROM is defined, the sender email address is set to MAILFROM. Otherwise mail is sent as "root (Cron Daemon)".

On the Debian GNU/Linux system, cron supports the **pam_env** module, and loads the environment specified by /etc/environment and /etc/security/pam_env.conf. It also reads locale information from /etc/default/locale. However, the PAM settings do **NOT** override the settings described above nor any settings in the `crontab` file itself.

By default, cron will send mail using the mail "Content-Type:" header of "text/plain" with the "charset=" parameter set to the charmap / codeset of the locale in which `crond(8)` is started up – i.e. either the default system locale, if no LC_* environment variables are set, or the locale specified by the LC_* environment variables (see `locale(7)`). You can use different character encodings for mailed cron job output by setting

the CONTENT_TYPE and CONTENT_TRANSFER_ENCODING variables in crontabs, to the correct values of the mail headers of those names.

The format of a cron command is very much the V7 standard, with a number of upward-compatible extensions. Each line has five time and date fields, followed by a command, followed by a newline character ('\n'). The system crontab (/etc/crontab) uses the same format, except that the username for the command is specified after the time and date fields and before the command. The fields may be separated by spaces or tabs. The maximum permitted length for the command field is 998 characters.

Commands are executed by *cron*(8) when the minute, hour, and month of year fields match the current time, *and* when at least one of the two day fields (day of month, or day of week) match the current time (see "Note" below). *cron*(8) examines cron entries once every minute. The time and date fields are:

field	allowed values
-----	-----
minute	0–59
hour	0–23
day of month	1–31
month	1–12 (or names, see below)
day of week	0–7 (0 or 7 is Sun, or use names)

A field may be an asterisk (*), which always stands for "first–last".

Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8–11 for an "hours" entry specifies execution at hours 8, 9, 10 and 11.

Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9", "0–4,8–12".

Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies skips of the number's value through the range. For example, "0–23/2" can be used in the hours field to specify command execution every other hour (the alternative in the V7 standard is "0,2,4,6,8,10,12,14,16,18,20,22"). Steps are also permitted after an asterisk, so if you want to say "every two hours", just use "*/2".

Names can also be used for the "month" and "day of week" fields. Use the first three letters of the particular day or month (case doesn't matter). Ranges or lists of names are not allowed.

The "sixth" field (the rest of the line) specifies the command to be run. The entire command portion of the line, up to a newline or % character, will be executed by /bin/sh or by the shell specified in the SHELL variable of the crontab file. Percent-signs (%) in the command, unless escaped with backslash (\), will be changed into newline characters, and all data after the first % will be sent to the command as standard input. There is no way to split a single command line onto multiple lines, like the shell's trailing "\".

Note: The day of a command's execution can be specified by two fields — day of month, and day of week. If both fields are restricted (i.e., don't start with *), the command will be run when *either* field matches the current time. For example,

"30 4 1,15 * 5" would cause a command to be run at 4:30 am on the 1st and 15th of each month, plus every Friday. One can, however, achieve the desired result by adding a test to the command (see the last example in EXAMPLE CRON FILE below).

Instead of the first five fields, one of eight special strings may appear:

string	meaning
-----	-----
@reboot	Run once, at startup.
@yearly	Run once a year, "0 0 1 1 *".
@annually	(same as @yearly)
@monthly	Run once a month, "0 0 1 * *".
@weekly	Run once a week, "0 0 * * 0".
@daily	Run once a day, "0 0 * * *".
@midnight	(same as @daily)

@hourly	Run once an hour, "0 * * * *".
---------	--------------------------------

Please note that startup, as far as @reboot is concerned, is the time when the *cron(8)* daemon startup. In particular, it may be before some system daemons, or other facilities, were startup. This is due to the boot order sequence of the machine.

EXAMPLE CRON FILE

The following lists an example of a user crontab file.

```
# use /bin/bash to run commands, instead of the default /bin/sh
SHELL=/bin/bash
# mail any output to 'paul', no matter whose crontab this is
MAILTO=paul
#
# run five minutes after midnight, every day
5 0 * * *    $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# run at 2:15pm on the first of every month — output mailed to paul
15 14 1 * *    $HOME/bin/monthly
# run at 10 pm on weekdays, annoy Joe
0 22 * * 1-5   mail -s "It's 10pm" joe%Joe,%%Where are your kids?%
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ..., everyday"
5 4 * * sun    echo "run at 5 after 4 every Sunday"
0 */4 1 * mon  echo "run every 4th hour on the 1st and on every Monday"
0 0 */2 * sun  echo "run at midn on every Sunday that's an uneven date"
# Run on every second Saturday of the month
0 4 8-14 * *   test $(date +\%u) -eq 6 && echo "2nd Saturday"
```

All the above examples run non-interactive programs. If you wish to run a program that interacts with the user's desktop you have to make sure the proper environment variable *DISPLAY* is set.

```
# Execute a program and run a notification every day at 10:00 am
0 10 * * * $HOME/bin/program | DISPLAY=:0 notify-send "Program run" "$(cat)"
```

EXAMPLE SYSTEM CRON FILE

The following lists the content of a regular system-wide crontab file. Unlike a user's crontab, this file has the username field, as used by */etc/crontab*.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
```

```
SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
```

```
# | | | |
# m h dom mon dow user  command
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Note that all the system-wide tasks will run, by default, from 6 am to 7 am. In the case of systems that are not powered on during that period of time, only the hourly tasks will be executed unless the defaults above are changed.

SEE ALSO

[cron\(8\)](#), [crontab\(1\)](#)

EXTENSIONS

When specifying day of week, both day 0 and day 7 will be considered Sunday. BSD and AT&T seem to disagree about this.

Lists and ranges are allowed to co-exist in the same field. "1-3,7-9" would be rejected by AT&T or BSD cron — they want to see "1-3" or "7,8,9" ONLY.

Ranges can include "steps", so "1-9/2" is the same as "1,3,5,7,9".

Months or days of the week can be specified by name.

Environment variables can be set in the crontab. In BSD or AT&T, the environment handed to child processes is basically the one from /etc/rc.

Command output is mailed to the crontab owner (BSD can't do this), can be mailed to a person other than the crontab owner (SysV can't do this), or the feature can be turned off and no mail will be sent at all (SysV can't do this either).

All of the '@' commands that can appear in place of the first five fields are extensions.

LIMITATIONS

The *cron* daemon runs with a defined timezone. It currently does not support per-user timezones. All the tasks: system's and user's will be run based on the configured timezone. Even if a user specifies the *TZ* environment variable in his *crontab* this will affect only the commands executed in the crontab, not the execution of the crontab tasks themselves.

POSIX specifies that the day of month and the day of week fields both need to match the current time if either of them is a *. However, this implementation only checks if the *first character* is a *. This is why "0 0 */2 * sun" runs every Sunday that's an uneven date while the POSIX standard would have it run every Sunday and on every uneven date.

The *crontab* syntax does not make it possible to define all possible periods one can imagine. For example, it is not straightforward to define the last weekday of a month. To have a task run in a time period that cannot be defined using *crontab* syntax, the best approach would be to have the program itself check the date and time information and continue execution only if the period matches the desired one.

If the program itself cannot do the checks then a wrapper script would be required. Useful tools that could be used for date analysis are *ncal* or *calendar*. For example, to run a program the last Saturday of every month you could use the following wrapper code:

```
0 4 * * Sat [ "$(date +\%e)" = "$(LANG=C ncal | sed -n 's/^Sa .* \([0-9]\)\+\)\ *$/\1/p'" ] && echo "Last Saturday" &&
```

DIAGNOSTICS

cron requires that each entry in a crontab end in a newline character. If the last entry in a crontab is missing a newline (i.e. terminated by EOF), cron will consider the crontab (at least partially) broken. A warning will be written to syslog.

AUTHOR

Paul Vixie <paul@vix.com> is the author of *cron* and original creator of this manual page. This page has also been modified for Debian by Steve Greenland, Javier Fernandez-Sanguino, Christian Kastner and Christian Pekeler.

NAME

cut – remove sections from each line of files

SYNOPSIS

cut *OPTION...* [*FILE*]...

DESCRIPTION

Print selected parts of lines from each FILE to standard output.

With no FILE, or when FILE is **-**, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-b, --bytes=LIST

select only these bytes

-c, --characters=LIST

select only these characters

-d, --delimiter=DELIM

use DELIM instead of TAB for field delimiter

-f, --fields=LIST

select only these fields; also print any line that contains no delimiter character, unless the **-s** option is specified

-n (ignored)

--complement

complement the set of selected bytes, characters or fields

-s, --only-delimited

do not print lines not containing delimiters

--output-delimiter=STRING

use STRING as the output delimiter the default is to use the input delimiter

-z, --zero-terminated

line delimiter is NUL, not newline

--help display this help and exit

--version

output version information and exit

Use one, and only one of **-b**, **-c** or **-f**. Each LIST is made up of one range, or many ranges separated by commas. Selected input is written in the same order that it is read, and is written exactly once. Each range is one of:

N N'th byte, character or field, counted from 1

N- from N'th byte, character or field, to end of line

N-M from N'th to M'th (included) byte, character or field

-M from first to M'th (included) byte, character or field

AUTHOR

Written by David M. Ihnat, David MacKenzie, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent

permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/cut>>
or available locally via: info '(coreutils) cut invocation'

NAME

date – print or set the system date and time

SYNOPSIS

```
date [OPTION]... [+FORMAT]
date [-u/--utc/--universal] [MMDDhhmm[[CC]YY][.ss]]
```

DESCRIPTION

Display the current time in the given FORMAT, or set the system date.

Mandatory arguments to long options are mandatory for short options too.

-d, --date=STRING

display time described by STRING, not 'now'

--debug

annotate the parsed date, and warn about questionable usage to stderr

-f, --file=DATEFILE

like **--date**; once for each line of DATEFILE

-I[FMT], --iso-8601[=FMT]

output date/time in ISO 8601 format. FMT='date' for date only (the default), 'hours', 'minutes', 'seconds', or 'ns' for date and time to the indicated precision. Example: 2006-08-14T02:34:56-06:00

-R, --rfc-email

output date and time in RFC 5322 format. Example: Mon, 14 Aug 2006 02:34:56 -0600

--rfc-3339=FMT

output date/time in RFC 3339 format. FMT='date', 'seconds', or 'ns' for date and time to the indicated precision. Example: 2006-08-14 02:34:56-06:00

-r, --reference=FILE

display the last modification time of FILE

-s, --set=STRING

set time described by STRING

-u, --utc, --universal

print or set Coordinated Universal Time (UTC)

--help display this help and exit

--version

output version information and exit

FORMAT controls the output. Interpreted sequences are:

%%	a literal %
%a	locale's abbreviated weekday name (e.g., Sun)
%A	locale's full weekday name (e.g., Sunday)
%b	locale's abbreviated month name (e.g., Jan)
%B	locale's full month name (e.g., January)
%c	locale's date and time (e.g., Thu Mar 3 23:05:25 2005)
%C	century; like %Y, except omit last two digits (e.g., 20)
%d	day of month (e.g., 01)
%D	date; same as %m/%d/%y
%e	day of month, space padded; same as %_d

%F	full date; like %+4Y-%m-%d
%g	last two digits of year of ISO week number (see %G)
%G	year of ISO week number (see %V); normally useful only with %V
%h	same as %b
%H	hour (00..23)
%I	hour (01..12)
%j	day of year (001..366)
%k	hour, space padded (0..23); same as %_H
%l	hour, space padded (1..12); same as %_I
%m	month (01..12)
%M	minute (00..59)
%n	a newline
%N	nanoseconds (00000000..99999999)
%p	locale's equivalent of either AM or PM; blank if not known
%P	like %p, but lower case
%q	quarter of year (1..4)
%r	locale's 12-hour clock time (e.g., 11:11:04 PM)
%R	24-hour hour and minute; same as %H:%M
%s	seconds since 1970-01-01 00:00:00 UTC
%S	second (00..60)
%t	a tab
%T	time; same as %H:%M:%S
%u	day of week (1..7); 1 is Monday
%U	week number of year, with Sunday as first day of week (00..53)
%V	ISO week number, with Monday as first day of week (01..53)
%w	day of week (0..6); 0 is Sunday
%W	week number of year, with Monday as first day of week (00..53)
%x	locale's date representation (e.g., 12/31/99)
%X	locale's time representation (e.g., 23:13:48)
%y	last two digits of year (00..99)
%Y	year
%z	+hhmm numeric time zone (e.g., -0400)
%:z	+hh:mm numeric time zone (e.g., -04:00)
%::z	+hh:mm:ss numeric time zone (e.g., -04:00:00)
%:::z	numeric time zone with : to necessary precision (e.g., -04, +05:30)
%Z	alphabetic time zone abbreviation (e.g., EDT)

By default, date pads numeric fields with zeroes. The following optional flags may follow '%':

– (hyphen) do not pad the field

- (underscore) pad with spaces
- 0 (zero) pad with zeros
- + pad with zeros, and put '+' before future years with >4 digits
- ^ use upper case if possible
- # use opposite case if possible

After any flags comes an optional field width, as a decimal number; then an optional modifier, which is either E to use the locale's alternate representations if available, or O to use the locale's alternate numeric symbols if available.

EXAMPLES

Convert seconds since the epoch (1970-01-01 UTC) to a date

```
$ date --date='@2147483647'
```

Show the time on the west coast of the US (use tzselect(1) to find TZ)

```
$ TZ='America/Los_Angeles' date
```

Show the local time for 9AM next Friday on the west coast of the US

```
$ date --date='TZ="America/Los_Angeles" 09:00 next Fri'
```

DATE STRING

The --date=STRING is a mostly free format human readable date string such as "Sun, 29 Feb 2004 16:21:42 -0800" or "2004-02-29 16:21:42" or even "next Thursday". A date string may contain items indicating calendar date, time of day, time zone, day of week, relative time, relative date, and numbers. An empty string indicates the beginning of the day. The date string format is more complex than is easily documented here but is fully described in the info documentation.

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/date>>

or available locally via: info '(coreutils) date invocation'

NAME

dd – convert and copy a file

SYNOPSIS

dd [*OPERAND*]...
dd *OPTION*

DESCRIPTION

Copy a file, converting and formatting according to the operands.

bs=BYTES

read and write up to BYTES bytes at a time (default: 512); overrides ibs and obs

cbs=BYTES

convert BYTES bytes at a time

conv=CONVS

convert the file as per the comma separated symbol list

count=N

copy only N input blocks

ibs=BYTES

read up to BYTES bytes at a time (default: 512)

if=FILE

read from FILE instead of stdin

iflag=FLAGS

read as per the comma separated symbol list

obs=BYTES

write BYTES bytes at a time (default: 512)

of=FILE

write to FILE instead of stdout

oflag=FLAGS

write as per the comma separated symbol list

seek=N skip N obs-sized blocks at start of output

skip=N skip N ibs-sized blocks at start of input

status=LEVEL

The LEVEL of information to print to stderr; 'none' suppresses everything but error messages, 'noxfer' suppresses the final transfer statistics, 'progress' shows periodic transfer statistics

N and BYTES may be followed by the following multiplicative suffixes: c=1, w=2, b=512, kB=1000, K=1024, MB=1000*1000, M=1024*1024, xM=M, GB=1000*1000*1000, G=1024*1024*1024, and so on for T, P, E, Z, Y. Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

Each CONV symbol may be:

ascii from EBCDIC to ASCII

ebcdic from ASCII to EBCDIC

ibm from ASCII to alternate EBCDIC

block pad newline-terminated records with spaces to cbs-size

unblock

replace trailing spaces in cbs-size records with newline

lcase change upper case to lower case

ucase change lower case to upper case

sparse try to seek rather than write all-NUL output blocks
swab swap every pair of input bytes
sync pad every input block with NULs to ibs-size; when used with block or unblock, pad with spaces rather than NULs
excl fail if the output file already exists
nocreat do not create the output file
notrunc do not truncate the output file
noerror continue after read errors
fdatasync physically write output file data before finishing
fsync likewise, but also write metadata

Each FLAG symbol may be:

append append mode (makes sense only for output; conv=notrunc suggested)
direct use direct I/O for data
directory fail unless a directory
dsync use synchronized I/O for data
sync likewise, but also for metadata
fullblock accumulate full blocks of input (iflag only)
nonblock use non-blocking I/O
noatime do not update access time
nocache Request to drop cache. See also oflag=sync
noctty do not assign controlling terminal from file
nofollow do not follow symlinks
count_bytes treat 'count=N' as a byte count (iflag only)
skip_bytes treat 'skip=N' as a byte count (iflag only)
seek_bytes treat 'seek=N' as a byte count (oflag only)

Sending a USR1 signal to a running 'dd' process makes it print I/O statistics to standard error and then resume copying.

Options are:

--help display this help and exit
--version output version information and exit

AUTHOR

Written by Paul Rubin, David MacKenzie, and Stuart Kemp.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/dd>>

or available locally via: info '(coreutils) dd invocation'

NAME

deluser, **delgroup** – remove a user or group from the system

SYNOPSIS

```
deluser [options] [--force] [--remove-home] [--remove-all-files] [--backup] [--backup-to DIR] user
deluser --group [options] group
delgroup [options] [--only-if-empty] group
deluser [options] user group
```

COMMON OPTIONS

```
[--quiet] [--system] [--help] [--version] [--conf FILE]
```

DESCRIPTION

deluser and **delgroup** remove users and groups from the system according to command line options and configuration information in */etc/deluser.conf* and */etc/adduser.conf*. They are friendlier front ends to the **userdel** and **groupdel** programs, removing the home directory as option or even all files on the system owned by the user to be removed, running a custom script, and other features. **deluser** and **delgroup** can be run in one of three modes:

Remove a normal user

If called with one non-option argument and without the **--group** option, **deluser** will remove a normal user.

By default, **deluser** will remove the user without removing the home directory, the mail spool or any other files on the system owned by the user. Removing the home directory and mail spool can be achieved using the **--remove-home** option.

The **--remove-all-files** option removes all files on the system owned by the user. Note that if you activate both options **--remove-home** will have no effect because all files including the home directory and mail spool are already covered by the **--remove-all-files** option.

If you want to backup all files before deleting them you can activate the **--backup** option which will create a file *username.tar(.gz|.bz2)* in the directory specified by the **--backup-to** option (defaulting to the current working directory). Both the remove and backup options can also be activated for default in the configuration file */etc/deluser.conf*. See **deluser.conf(5)** for details.

If you want to remove the root account (uid 0), then use the **--force** parameter; this may prevent to remove the root user by accident.

If the file **/usr/local/sbin/deluser.local** exists, it will be executed after the user account has been removed in order to do any local cleanup. The arguments passed to **deluser.local** are:
username uid gid home-directory

Remove a group

If **deluser** is called with the **--group** option, or **delgroup** is called, a group will be removed.

Warning: The primary group of an existing user cannot be removed.

If the option **--only-if-empty** is given, the group won't be removed if it has any members left.

Remove a user from a specific group

If called with two non-option arguments, **deluser** will remove a user from a specific group.

OPTIONS

--conf FILE

Use FILE instead of the default files */etc/deluser.conf* and */etc/adduser.conf*

--group

Remove a group. This is the default action if the program is invoked as *delgroup*.

--help Display brief instructions.**--quiet**

Suppress progress messages.

--system

Only delete if user/group is a system user/group. This avoids accidentally deleting non-system users/groups. Additionally, if the user does not exist, no error value is returned. This option is mainly for use in Debian package maintainer scripts.

--only-if-empty

Only remove if no members are left.

--backup

Backup all files contained in the userhome and the mailspool-file to a file named */\$user.tar.bz2* or */\$user.tar.gz*.

--backup-to

Place the backup files not in / but in the directory specified by this parameter. This implicitly sets --backup also.

--remove-home

Remove the home directory of the user and its mailspool. If --backup is specified, the files are deleted after having performed the backup.

--remove-all-files

Remove all files from the system owned by this user. Note: --remove-home does not have an effect any more. If --backup is specified, the files are deleted after having performed the backup.

--version

Display version and copyright information.

RETURN VALUE

- 0** The action was successfully executed.
- 1** The user to delete was not a system account. No action was performed.
- 2** There is no such user. No action was performed.
- 3** There is no such group. No action was performed.
- 4** Internal error. No action was performed.
- 5** The group to delete is not empty. No action was performed.
- 6** The user does not belong to the specified group. No action was performed.
- 7** You cannot remove a user from its primary group. No action was performed.
- 8** The required perl-package 'perl modules' is not installed. This package is required to perform the requested actions. No action was performed.
- 9** For removing the root account the parameter "--force" is required. No action was performed.

FILES

/etc/deluser.conf Default configuration file for deluser and delgroup

/usr/local/sbin/deluser.local

Optional custom add-ons.

SEE ALSO

adduser(8), deluser.conf(5), groupdel(8), userdel(8)

COPYRIGHT

Copyright (C) 2000 Roland Bauerschmidt. Modifications (C) 2004 Marc Haber and Joerg Hoh. This man-page and the deluser program are based on adduser which is:

Copyright (C) 1997, 1998, 1999 Guy Maor.

Copyright (C) 1995 Ted Hajek, with a great deal borrowed from the original Debian **adduser**

Copyright (C) 1994 Ian Murdock. **deluser** is free software; see the GNU General Public Licence version 2 or later for copying conditions. There is *no* warranty.

NAME

depmod – Generate modules.dep and map files.

SYNOPSIS

```
depmod [-b basedir] [-e] [-E Module.symvers] [-F System.map] [-n] [-v] [-A] [-P prefix] [-w] [version]
depmod [-e] [-E Module.symvers] [-F System.map] [-n] [-v] [-P prefix] [-w] [version] [filename...]
```

DESCRIPTION

Linux kernel modules can provide services (called "symbols") for other modules to use (using one of the EXPORT_SYMBOL variants in the code). If a second module uses this symbol, that second module clearly depends on the first module. These dependencies can get quite complex.

depmod creates a list of module dependencies by reading each module under /lib/modules/*version* and determining what symbols it exports and what symbols it needs. By default, this list is written to modules.dep, and a binary hashed version named modules.dep.bin, in the same directory. If filenames are given on the command line, only those modules are examined (which is rarely useful unless all modules are listed). **depmod** also creates a list of symbols provided by modules in the file named modules.symbols and its binary hashed version, modules.symbols.bin. Finally, **depmod** will output a file named modules.devname if modules supply special device names (devname) that should be populated in /dev on boot (by a utility such as systemd-tmpfiles).

If a *version* is provided, then that kernel version's module directory is used rather than the current kernel version (as returned by **uname -r**).

OPTIONS**-a, --all**

Probe all modules. This option is enabled by default if no file names are given in the command-line.

-A, --quick

This option scans to see if any modules are newer than the modules.dep file before any work is done: if not, it silently exits rather than regenerating the files.

-b *basedir*, --basedir *basedir*

If your modules are not currently in the (normal) directory /lib/modules/*version*, but in a staging area, you can specify a *basedir* which is prepended to the directory name. This *basedir* is stripped from the resulting modules.dep file, so it is ready to be moved into the normal location. Use this option if you are a distribution vendor who needs to pre-generate the meta-data files rather than running depmod again later.

-C, --config *file or directory*

This option overrides the default configuration directory at /etc/depmod.d/.

-e, --errsyms

When combined with the **-F** option, this reports any symbols which a module needs which are not supplied by other modules or the kernel. Normally, any symbols not provided by modules are assumed to be provided by the kernel (which should be true in a perfect world), but this assumption can break especially when additionally updated third party drivers are not correctly installed or were built incorrectly.

-E, --symvers

When combined with the **-e** option, this reports any symbol versions supplied by modules that do not match with the symbol versions provided by the kernel in its Module.symvers. This option is mutually incompatible with **-F**.

-F, --filesyms *System.map*

Supplied with the System.map produced when the kernel was built, this allows the **-e** option to report unresolved symbols. This option is mutually incompatible with **-E**.

-h, --help

Print the help message and exit.

-n, --show, --dry-run

This sends the resulting modules.dep and the various map files to standard output rather than writing them into the module directory.

-P

Some architectures prefix symbols with an extraneous character. This specifies a prefix character (for example '_') to ignore.

-v, --verbose

In verbose mode, **depmod** will print (to stdout) all the symbols each module depends on and the module's file name which provides that symbol.

-V, --version

Show version of program and exit. See below for caveats when run on older kernels.

-w

Warn on duplicate dependencies, aliases, symbol versions, etc.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Portions Copyright Jon Masters, and others.

SEE ALSO

depmod.d(5), modprobe(8), modules.dep(5)

AUTHORS

Jon Masters <jcm@jonmasters.org>
Developer

Robby Workman <rworkman@slackware.com>
Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>
Developer

NAME

df – report file system disk space usage

SYNOPSIS

df [*OPTION*]... [*FILE*]...

DESCRIPTION

This manual page documents the GNU version of **df**. **df** displays the amount of disk space available on the file system containing each file name argument. If no file name is given, the space available on all currently mounted file systems is shown. Disk space is shown in 1K blocks by default, unless the environment variable `POSIXLY_CORRECT` is set, in which case 512-byte blocks are used.

If an argument is the absolute file name of a disk device node containing a mounted file system, **df** shows the space available on that file system rather than on the file system containing the device node. This version of **df** cannot show the space available on unmounted file systems, because on most kinds of systems doing so requires very nonportable intimate knowledge of file system structures.

OPTIONS

Show information about the file system on which each FILE resides, or all file systems by default.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

include pseudo, duplicate, inaccessible file systems

-B, --block-size=SIZE

scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

-h, --human-readable

print sizes in powers of 1024 (e.g., 1023M)

-H, --si

print sizes in powers of 1000 (e.g., 1.1G)

-i, --inodes

list inode information instead of block usage

-k like --block-size=1K

-l, --local

limit listing to local file systems

--no-sync

do not invoke sync before getting usage info (default)

--output=[FIELD_LIST]

use the output format defined by FIELD_LIST, or print all fields if FIELD_LIST is omitted.

-P, --portability

use the POSIX output format

--sync invoke sync before getting usage info

--total elide all entries insignificant to available space, and produce a grand total

-t, --type=TYPE

limit listing to file systems of type TYPE

-T, --print-type

print file system type

-x, --exclude-type=TYPE

limit listing to file systems not of type TYPE

-v (ignored)

--help display this help and exit

--version

output version information and exit

Display values are in units of the first available SIZE from **--block-size**, and the DF_BLOCK_SIZE, BLOCK_SIZE and BLOCKSIZE environment variables. Otherwise, units default to 1024 bytes (or 512 if POSIXLY_CORRECT is set).

The SIZE argument is an integer and optional unit (example: 10K is 10*1024). Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000). Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

FIELD_LIST is a comma-separated list of columns to be included. Valid field names are: 'source', 'fstype', 'itotal', 'iused', 'iavail', 'ipcent', 'size', 'used', 'avail', 'pcent', 'file' and 'target' (see info page).

AUTHOR

Written by Torbjorn Granlund, David MacKenzie, and Paul Eggert.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/df>>
or available locally via: info '(coreutils) df invocation'

NAME

diff – compare files line by line

SYNOPSIS

diff [*OPTION*]... *FILES*

DESCRIPTION

Compare *FILES* line by line.

Mandatory arguments to long options are mandatory for short options too.

--normal

output a normal diff (the default)

-q, --brief

report only when files differ

-s, --report-identical-files

report when two files are the same

-c, -C NUM, --context[=NUM]

output NUM (default 3) lines of copied context

-u, -U NUM, --unified[=NUM]

output NUM (default 3) lines of unified context

-e, --ed

output an ed script

-n, --rcs

output an RCS format diff

-y, --side-by-side

output in two columns

-W, --width=NUM

output at most NUM (default 130) print columns

--left-column

output only the left column of common lines

--suppress-common-lines

do not output common lines

-p, --show-c-function

show which C function each change is in

-F, --show-function-line=RE

show the most recent line matching RE

--label LABEL

use LABEL instead of file name and timestamp (can be repeated)

-t, --expand-tabs

expand tabs to spaces in output

-T, --initial-tab

make tabs line up by prepending a tab

--tabsize=NUM

tab stops every NUM (default 8) print columns

--suppress-blank-empty

suppress space or tab before empty output lines

-l, --paginate

pass output through 'pr' to paginate it

-r, --recursive
recursively compare any subdirectories found

--no-dereference
don't follow symbolic links

-N, --new-file
treat absent files as empty

--unidirectional-new-file
treat absent first files as empty

--ignore-file-name-case
ignore case when comparing file names

--no-ignore-file-name-case
consider case when comparing file names

-x, --exclude=PAT
exclude files that match PAT

-X, --exclude-from=FILE
exclude files that match any pattern in FILE

-S, --starting-file=FILE
start with FILE when comparing directories

--from-file=FILE1
compare FILE1 to all operands; FILE1 can be a directory

--to-file=FILE2
compare all operands to FILE2; FILE2 can be a directory

-i, --ignore-case
ignore case differences in file contents

-E, --ignore-tab-expansion
ignore changes due to tab expansion

-Z, --ignore-trailing-space
ignore white space at line end

-b, --ignore-space-change
ignore changes in the amount of white space

-w, --ignore-all-space
ignore all white space

-B, --ignore-blank-lines
ignore changes where lines are all blank

-I, --ignore-matching-lines=RE
ignore changes where all lines match RE

-a, --text
treat all files as text

--strip-trailing-cr
strip trailing carriage return on input

-D, --ifdef=NAME
output merged file with '#ifdef NAME' diffs

--GTYPE-group-format=GFMT
format GTYPE input groups with GFMT

--line-format=LFMT

format all input lines with LFMT

--LTYPE-line-format=LFMT

format LTYPE input lines with LFMT

These format options provide fine-grained control over the output
of diff, generalizing **-D**/**--ifdef**.

LTYPE is 'old', 'new', or 'unchanged'.

GTYPE is LTYPE or 'changed'.

GFMT (only) may contain:

%< lines from FILE1

%> lines from FILE2

%= lines common to FILE1 and FILE2

%[-][WIDTH][.[PREC]]{doX}LETTER
printf-style spec for LETTER

LETTERS are as follows for new group, lower case for old group:

F first line number

L last line number

N number of lines = L-F+1

E F-1

M L+1

%**(A=B?T:E)**

if A equals B then T else E

LFMT (only) may contain:

%L contents of line

%l contents of line, excluding any trailing newline

%[-][WIDTH][.[PREC]]{doX}n
printf-style spec for input line number

Both GFMT and LFMT may contain:

%% %

%c'C' the single character C

%c'\OOO'
the character with octal code OOO

C the character C (other characters represent themselves)

-d, --minimal

try hard to find a smaller set of changes

--horizon-lines=NUM

keep NUM lines of the common prefix and suffix

--speed-large-files

assume large files and many scattered small changes

--color[=WHEN]

color output; WHEN is 'never', 'always', or 'auto'; plain **--color** means **--color='auto'**

--palette=PALETTE

the colors to use when **--color** is active; PALETTE is a colon-separated list of terminfo capabilities

--help display this help and exit**-v, --version**

output version information and exit

FILES are 'FILE1 FILE2' or 'DIR1 DIR2' or 'DIR FILE' or 'FILE DIR'. If **--from-file** or **--to-file** is given, there are no restrictions on FILE(s). If a FILE is '−', read standard input. Exit status is 0 if inputs are the same, 1 if different, 2 if trouble.

AUTHOR

Written by Paul Eggert, Mike Haertel, David Hayes, Richard Stallman, and Len Tower.

REPORTING BUGS

Report bugs to: bug-diffutils@gnu.org

GNU diffutils home page: <<https://www.gnu.org/software/diffutils/>>

General help using GNU software: <<https://www.gnu.org/gethelp/>>

COPYRIGHT

Copyright © 2021 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`wdiff(1)`, `cmp(1)`, `diff3(1)`, `sdiff(1)`, `patch(1)`

The full documentation for **diff** is maintained as a Texinfo manual. If the **info** and **diff** programs are properly installed at your site, the command

info diff

should give you access to the complete manual.

NAME

diff3 – compare three files line by line

SYNOPSIS

diff3 [*OPTION*]... *MYFILE OLDFILE YOURFILE*

DESCRIPTION

Compare three files line by line.

Mandatory arguments to long options are mandatory for short options too.

-A, --show-all

output all changes, bracketing conflicts

-e, --ed

output ed script incorporating changes from *OLDFILE* to *YOURFILE* into *MYFILE*

-E, --show-overlap

like **-e**, but bracket conflicts

-3, --easy-only

like **-e**, but incorporate only nonoverlapping changes

-x, --overlap-only

like **-e**, but incorporate only overlapping changes

-X like **-x**, but bracket conflicts

-i append 'w' and 'q' commands to ed scripts

-m, --merge

output actual merged file, according to **-A** if no other options are given

-a, --text

treat all files as text

--strip-trailing-cr

strip trailing carriage return on input

-T, --initial-tab

make tabs line up by prepending a tab

--diff-program=PROGRAM

use *PROGRAM* to compare files

-L, --label=LABEL

use *LABEL* instead of file name (can be repeated up to three times)

--help display this help and exit

-v, --version

output version information and exit

The default output format is a somewhat human-readable representation of the changes.

The **-e**, **-E**, **-x**, **-X** (and corresponding long) options cause an ed script to be output instead of the default.

Finally, the **-m** (**--merge**) option causes diff3 to do the merge internally and output the actual merged file. For unusual input, this is more robust than using ed.

If a FILE is '–', read standard input. Exit status is 0 if successful, 1 if conflicts, 2 if trouble.

AUTHOR

Written by Randy Smith.

REPORTING BUGS

Report bugs to: bug-diffutils@gnu.org

GNU diffutils home page: <<https://www.gnu.org/software/diffutils/>>

General help using GNU software: <<https://www.gnu.org/gethelp/>>

COPYRIGHT

Copyright © 2021 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`cmp(1)`, `diff(1)`, `sdiff(1)`

The full documentation for **diff3** is maintained as a Texinfo manual. If the **info** and **diff3** programs are properly installed at your site, the command

info diff3

should give you access to the complete manual.

NAME

Dislocate – disconnect and reconnect processes

SYNOPSIS

dislocate [*program args...*]

INTRODUCTION

Dislocate allows processes to be disconnected and reconnected to the terminal. Possible uses:

- You can disconnect a process from a terminal at work and reconnect from home, to continue working.
- After having your line be dropped due to noise, you can get back to your process without having to restart it from scratch.
- If you have a problem that you would like to show someone, you can set up the scenario at your own terminal, disconnect, walk down the hall, and reconnect on another terminal.
- If you are in the middle of a great game (or whatever) that does not allow you to save, and someone else kicks you off the terminal, you can disconnect, and reconnect later.

USAGE

When run with no arguments, **Dislocate** tells you about your disconnected processes and lets you reconnect to one. Otherwise, **Dislocate** runs the named program along with any arguments.

By default, ^] is an escape that lets you talk to **Dislocate** itself. At that point, you can disconnect (by pressing ^D) or suspend **Dislocate** (by pressing ^Z).

Any Tcl or Expect command is also acceptable at this point. For example, to insert the contents of a the file /etc/motd as if you had typed it, say:

```
send -i $out [exec cat /etc/motd]
```

To send the numbers 1 to 100 in response to the prompt "next #", say:

```
for {set i 0} {$i<100} {incr i} {
    expect -i $in "next #"
    send -i $out "$i\r"
}
```

Scripts can also be prepared and sourced in so that you don't have to type them on the spot.

Dislocate is actually just a simple **Expect** script. Feel free to make it do what you want it to do or just use **Expect** directly, without going through **Dislocate**. **Dislocate** understands a few special arguments. These should appear before any program name. Each should be separated by whitespace. If the arguments themselves takes arguments, these should also be separated by whitespace.

The **-escape** flag sets the escape to whatever follows. The default escape is ^].

CAVEATS

This program was written by the author as an exercise to show that communicating with disconnected processes is easy. There are many features that could be added, but that is not the intent of this program.

SEE ALSO

Tcl(3), libexpect(3)

"Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs" by Don Libes, O'Reilly and Associates, January 1995.

AUTHOR

Don Libes, National Institute of Standards and Technology

NAME

dmesg – print or control the kernel ring buffer

SYNOPSIS

dmesg [options]

dmesg --clear

dmesg --read-clear [options]

dmesg --console-level *level*

dmesg --console-on

dmesg --console-off

DESCRIPTION

dmesg is used to examine or control the kernel ring buffer.

The default action is to display all messages from the kernel ring buffer.

OPTIONS

The **--clear**, **--read-clear**, **--console-on**, **--console-off**, and **--console-level** options are mutually exclusive.

-C, --clear

Clear the ring buffer.

-c, --read-clear

Clear the ring buffer after first printing its contents.

-D, --console-off

Disable the printing of messages to the console.

-d, --show-delta

Display the timestamp and the time delta spent between messages. If used together with **--notime** then only the time delta without the timestamp is printed.

-E, --console-on

Enable printing messages to the console.

-e, --reltime

Display the local time and the delta in human-readable format. Be aware that conversion to the local time could be inaccurate (see **-T** for more details).

-F, --file *file*

Read the syslog messages from the given *file*. Note that **-F** does not support messages in kmsg format. The old syslog format is supported only.

-f, --facility *list*

Restrict output to the given (comma-separated) *list* of facilities. For example:

dmesg --facility=daemon

will print messages from system daemons only. For all supported facilities see the **--help** output.

-H, --human

Enable human-readable output. See also **--color**, **--reltime** and **--nopager**.

-J, --json

Use JSON output format. The time output format is in "sec.usec" format only, log priority level is not decoded by default (use **--decode** to split into facility and priority), the other options to control the output format or time format are silently ignored.

-k, --kernel

Print kernel messages.

-L, --color[=when]

Colorize the output. The optional argument *when* can be **auto**, **never** or **always**. If the *when* argument is omitted, it defaults to **auto**. The colors can be disabled; for the current built-in default see the **--help** output. See also the **COLORS** section below.

-l, --level list

Restrict output to the given (comma-separated) *list* of levels. For example:

```
dmesg --level=err,warn
```

will print error and warning messages only. For all supported levels see the **--help** output.

-n, --console-level level

Set the *level* at which printing of messages is done to the console. The *level* is a level number or abbreviation of the level name. For all supported levels see the **--help** output.

For example, **-n 1** or **-n emerg** prevents all messages, except emergency (panic) messages, from appearing on the console. All levels of messages are still written to */proc/kmsg*, so **syslogd(8)** can still be used to control exactly where kernel messages appear. When the **-n** option is used, **dmesg** will *not* print or clear the kernel ring buffer.

--noescape

The unprintable and potentially unsafe characters (e.g., broken multi-byte sequences, terminal controlling chars, etc.) are escaped in format **\x<hex>** for security reason by default. This option disables this feature at all. It's usable for example for debugging purpose together with **--raw**. Be careful and don't use it by default.

-P, --nopager

Do not pipe output into a pager. A pager is enabled by default for **--human** output.

-p, --force-prefix

Add facility, level or timestamp information to each line of a multi-line message.

-r, --raw

Print the raw message buffer, i.e., do not strip the log-level prefixes, but all unprintable characters are still escaped (see also **--noescape**).

Note that the real raw format depends on the method how **dmesg** reads kernel messages. The */dev/kmsg* device uses a different format than **syslog(2)**. For backward compatibility, **dmesg** returns data always in the **syslog(2)** format. It is possible to read the real raw data from */dev/kmsg* by, for example, the command 'dd if=/dev/kmsg iflag=nonblock'.

-S, --syslog

Force **dmesg** to use the **syslog(2)** kernel interface to read kernel messages. The default is to use **/dev/kmsg** rather than **syslog(2)** since kernel 3.5.0.

-s, --buffer-size *size*

Use a buffer of *size* to query the kernel ring buffer. This is 16392 by default. (The default kernel syslog buffer size was 4096 at first, 8192 since 1.3.54, 16384 since 2.1.113.) If you have set the kernel buffer to be larger than the default, then this option can be used to view the entire buffer.

-T, --ctime

Print human-readable timestamps.

Be aware that the timestamp could be inaccurate! The **time** source used for the logs is **not updated after system SUSPEND/RESUME**. Timestamps are adjusted according to current delta between boottime and monotonic clocks, this works only for messages printed after last resume.

--since *time*

Display record since the specified time. The time is possible to specify in absolute way as well as by relative notation (e.g. '1 hour ago'). Be aware that the timestamp could be inaccurate and see **--ctime** for more details.

--until *time*

Display record until the specified time. The time is possible to specify in absolute way as well as by relative notation (e.g. '1 hour ago'). Be aware that the timestamp could be inaccurate and see **--ctime** for more details.

-t, --notime

Do not print kernel's timestamps.

--time-format *format*

Print timestamps using the given *format*, which can be **ctime**, **reltime**, **delta** or **iso**. The first three formats are aliases of the time-format-specific options. The **iso** format is a **dmesg** implementation of the ISO-8601 timestamp format. The purpose of this format is to make the comparing of timestamps between two systems, and any other parsing, easy. The definition of the **iso** timestamp is:
YYYY-MM-DD<T>HH:MM:SS,<microseconds><+><timezone offset from UTC>.

The **iso** format has the same issue as **ctime**: the time may be inaccurate when a system is suspended and resumed.

-u, --userspace

Print userspace messages.

-w, --follow

Wait for new messages. This feature is supported only on systems with a readable **/dev/kmsg** (since kernel 3.5.0).

-W, --follow-new

Wait and print only new messages.

-x, --decode

Decode facility and level (priority) numbers to human-readable prefixes.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

COLORS

The output colorization is implemented by **terminal-colors.d(5)** functionality. Implicit coloring can be disabled by an empty file

/etc/terminal-colors.d/dmesg.disable

for the **dmesg** command or for all tools by

/etc/terminal-colors.d/disable

The user-specific *\$XDG_CONFIG_HOME/terminal-colors.d* or *\$HOME/.config/terminal-colors.d* overrides the global setting.

Note that the output colorization may be enabled by default, and in this case *terminal-colors.d* directories do not have to exist yet.

The logical color names supported by **dmesg** are:

subsys

The message sub-system prefix (e.g., "ACPI:").

time

The message timestamp.

timebreak

The message timestamp in short ctime format in **--reltime** or **--human** output.

alert

The text of the message with the alert log priority.

crit

The text of the message with the critical log priority.

err

The text of the message with the error log priority.

warn

The text of the message with the warning log priority.

segfault

The text of the message that inform about segmentation fault.

EXIT STATUS

dmesg can fail reporting permission denied error. This is usually caused by **dmesg_restrict** kernel setting, please see **syslog(2)** for more details.

AUTHORS

Karel Zak <kzak@redhat.com>

dmesg was originally written by Theodore Ts'o <tytso@athena.mit.edu>.

SEE ALSO

terminal-colors.d(5), syslogd(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **dmesg** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

dmraid(8) - Linux man page

Name

dmraid - discover, configure and activate software (ATA)RAID

Synopsis

```
dmraid {-a|--activate} {y|n|yes|no}
[-d|--debug]... [-v|--verbose]... [-i|--ignorelocking]
[-f|--format FORMAT[,FORMAT...]]]
[{-P|--partchar} CHAR]
[-p|--no_partitions]
[-Z|--rm_partitions]
[--separator SEPARATOR]
[-t|--test]
[RAID-set...]
```

```
dmraid {-b|--block_devices}
[-c|--display_columns][FIELD[,FIELD...]]...
[-d|--debug]... [-v|--verbose]...
[--separator SEPARATOR]
[device-path...]
```

```
dmraid {-h|--help}
```

```
dmraid {-l|--list_formats}
[-d|--debug]... [-v|--verbose]...
```

```
dmraid {-n|--native_log}
[-d|--debug]... [-v|--verbose]... [-i|--ignorelocking]
[-f|--format FORMAT[,FORMAT...]]]
[--separator SEPARATOR]
[device-path...]
```

```
dmraid {-R| --rebuild}
```

```
RAID-set
[device-path]
```

```
dmraid {-x| --remove}
[RAID-set]
```

```
dmraid -f FORMAT-handler
{-C| --create} set --type raidlevel
```

[--size=setsize --strip stridesize]
--disk "device-path, device-path [, device-path ...]"

dmraid [-f|--format FORMAT-handler] -S|--spare [RAID-set] -M|--media "device-path"

dmraid {-r|--raid_devices}

[-c|--display_columns][FIELD[,FIELD...]]...
[-d|--debug]... [-v|--verbose]... [-i|--ignorelocking]
[-D|--dump_metadata]
[-f|--format FORMAT[,FORMAT...]]
[--separator SEPARATOR]
[device-path...]

dmraid {-r|--raid_devices}

[-d|--debug]... [-v|--verbose]... [-i|--ignorelocking]
[-E|--erase_metadata]
[-f|--format FORMAT[,FORMAT...]]
[--separator SEPARATOR]
[device-path...]

dmraid {-s|--sets}...[a|i|active|inactive]

[-c|--display_columns][FIELD[,FIELD...]]...
[-d|--debug]... [-v|--verbose]... [-i|--ignorelocking]
[-f|--format FORMAT[,FORMAT...]]
[-g|--display_group]
[--separator SEPARATOR]
[RAID-set...]

dmraid {-V|--version}

Description

dmraid discovers block and software RAID devices (eg, ATARAID) by using multiple different metadata format handlers which support various formats (eg, Highpoint 37x series). It offers activating RAID sets made up by 2 or more discovered RAID devices, display properties of devices and sets (see option **-I** for supported metadata formats). Block device access to activated RAID sets occurs via device-mapper nodes /dev/mapper/RaidSetName. RaidSetName starts with the format name (see **-I** option) which can be used to access all RAID sets of a specific format easily with certain options (eg, **-a** below).

Options

-a, **--activate** {y|n} [RAID set...]

Activates or deactivates all or particular software RAID set. In case metadata format handlers are chosen with **-f**, only RAID sets with such **format(s)** can be activated or

deactivated. Useful if devices have multiple metadata signatures. When activating RAID sets, **-p** disables the activation of partitions on them, and **-Z** will make dmraid tell the kernel to remove the partitions from the disks underlying the set, ie if sda is part of the set, remove sda1, sda2, etc. This prevents applications from directly accessing the disks bypassing dmraid. RAID set names given on command line don't need to be fully specified (eg, "dmraid -ay sil" would activate all discovered Silicon Image Medley RAID sets).

{-b|--block_devices} [device-path...]

List all or particular discovered block devices with their properties (size, serial number). Add **-c** to display block device names only and **-cc** for CSV column output of block device properties. See description of **-c** below for FIELD identifiers.

{-d|--debug}...

Enable debugging output. Option can be given multiple times increasing the debug output level.

{-c|--display_columns}[FIELD[,FIELD...]]...

Display properties of block devices, RAID sets and devices in **column**(s). Optional list specifying which FIELDS to display.

For **-b**:

d[evpath]|p[ath], sec[tors]|si[ze], ser[ialnumber].

For **-r**:

de[vpath]|p[ath], f[ormat], r[aidname], t[ype], st[atus], se[ctors]|si[ze], da[taoffset]|o[ffset].

For **-s**:

f[ormat], r[aidname], t[ype], sta[tus], str[ide], se[ctors]|si[ze], su[bsets], d[evices], sp[ares].

{-f|--format FORMAT[,FORMAT...]]

Use metadata format **handler**(s) to discover RAID devices. See **-I** for a list of supported format handler names. This is useful to select particular formats in case multiple metadata signatures are found on a device. A comma separated list of format names can be specified which may not contain white space.

{-h|--help}

Display help text.

{-i|--ignorelocking}

Don't take out any locks. Useful in early boot where no read/write access to /var is available.

{-l|--list_formats}

List all available metadata format handlers with their names and descriptions.

Supported RAID levels are listed in parenthesis:

S: Span (concatination)

0: RAID0 (stripe)

1: RAID1 (mirror)

10: RAID10 (mirror on top of stripes)

01: RAID10 (stripe on top of mirrors) Note: Intel OROM displays this as RAID10

{-n|--native_log} [device-path...]

Display metadata in native, vendor-specific format. In case a metadata format handler is chosen with **-f** only RAID devices with such format will be displayed in native format. If **device-path(s)** is/are given on the command line, native metadata output is restricted to those listed.

[{-P|--partchar} CHAR]

Use CHAR as the separator between the device name and the partition number.

{-R|--rebuild} RAID-set [device-path]

Rebuild raid array after a drive has failed and a new drive is added. For Intel chipset based systems, there are two methods in which a new drive is added to the system.

1. Using OROM to identify a new drive During system reboot, enter OROM and mark the new drive as the rebuild drive.

After booting to the OS, use the dmraid command to rebuild.

Example: dmraid -R raid_set

2. Using dmraid to identify a new drive Boot to the OS and use the dmraid command with the new drive as the second parameter.

Example: dmraid -R raid_set /dev/sdc

3. Using hot spare drive Mark a drive as hot spare using the "dmraid -f isw -S" command. Then use the dmraid command to start the rebuild.

Example: dmraid -R raid_set

{-x|--remove} [RAID-set]

Delete one or all existing software RAID devices from the metadata.

-f FORMAT-handler **{-C|--create}** --type raidlevel [--size=setsize --strip=stripsize] --disk "device-path, device-path [,device-path]"

Delete one or all existing Configure a software RAID device and store the configuration data in a group of hard drive devices consisting of this array. This command requires the following options:

-f FORMAT-handler

metadata format (see "dmraid -l")

--type digit[digit...]

specify the raid level of the software RAID set.

0: raid0

1: raid1

5: raid5

01: raid01 (isw raid10)

--size: [digits[k|K|m|M|g|G][b|B]]

specify the size of the RAID set. The number is an integer followed by [kKmMgG] and/or [bB].

b: byte (default)

B: block (512 bytes)

K or k: on the base of 1024

m or M: on the base of 1024*1024

g or G: on the base of 1024*1024*1024

If this option is missing, it's set to the default value pre-configured by the vendor.

Note that different vendors may apply different constraints on the granularity of the size or the minimal value.

--strip: [digits[k|K|m|M|g|G][b|B]]

specify the strip size of a RAID1, RAID5, and RAID10 RAID set (as above)

--disk: device-path[{| }device-path...]

specify the array of the hard drives, e.g. /dev/sda.

-f FORMAT-handler -S -M device-path

-S -M device-path

This command adds hot spare support for one or more RAID sets.

1. When used with a format handler, which supports hot spare sets (e.g. isw), a hot spare is marked to be used when rebuilding any RAID set of that format. 2. When used when specifying a RAID set, the drive is added to that RAID set and will be used only to rebuild that set. Note: If the specified name does not match an existing RAID-set, a set with the new name will be created.

{-r|--raid_devices} [device-path...]

List all discovered RAID devices with format, RAID level, sectors used and data offset into the device. In case a metadata format handler is chosen with **-f**, only RAID devices with such format can be discovered. Useful if devices have multiple metadata signatures. If **-D** is added to **-r** the RAID metadata gets dumped into a subdirectory named dmraid.format_name (eg. format_name = isw) in files named devicename.dat. The byte offset where the metadata is located on the device is written into files named devicename.offset and the size of the device in sectors into files named devicename.size.

If **-E** is added to **-r** the RAID metadata on the devices gets conditionally erased.

Useful to erase old metadata after new one of different type has been stored on a device in order to avoid discovering both. If you enter **-E** option **-D** will be enforced in order to have a fallback in case the wrong metadata got erased. Manual copying back onto the device is needed to recover from erasing the wrong metadata using the

dumped files devicename_formatname.dat and devicename_formatname.offset. Eg, to restore all *.dat files in the working directory to the respective devices:

```
for f in *.dat
do
dd if=$f of=/dev/${f%%.dat} \
seek='cat ${f%%dat}offset' bs=1

done
```

If **device-path**(s) is/are given on the command line, the above actions are restricted to those listed. Add **-c** to display RAID device names only and **-cc** for CSV column output of RAID device properties. See description of **-c** above for FIELD identifiers.

--separator SEPARATOR

Use SEPARATOR as a delimiter for all options taking or displaying lists.

-s... [a|i] [RAID-set...]

Display properties of RAID sets. Multiple RAID set names can be given on the command line which don't need to be fully specified (eg, "dmraid -s hpt" would display all discovered Highpoint RAID sets). Enter **-s** twice to display RAID subsets too. Add **-c** to display names of RAID sets only, **-cc** for CSV column output of RAID set properties and **-ccc** for inclusion of block devices in the listing. Doesn't imply **-s -s** to show RAID subsets (implied for group sets, e.g. isw). Add **-g** to include information about group RAID sets (as with Intel Software RAID) in the listing. See description of **-c** above for FIELD identifiers. Note: Size is given in sectors (not bytes).

[-v|--verbose]...

Enable verbose runtime information output. Option can be given multiple times increasing the verbosity level.

Examples

"dmraid -l" lists all supported metadata formats with their names along with some descriptive information, eg:

```
hpt37x : (+) Highpoint HPT37X
hpt45x : (+) Highpoint HPT45X
isw : (+) Intel Software RAID
lsi : (0) LSI Logic MegaRAID
nvidia : (+) NVidia RAID
pdc : (+) Promise FastTrack
sil : (+) Silicon Image(tm) Medley(tm)
via : (+) VIA Software RAID
dos : (+) DOS partitions on SW RAIDs
(0): Discover, (+): Discover+Activate
"dmraid -ay" activates all software RAID sets discovered.
"dmraid -an" deactivates all active software RAID sets which are not open (eg, mounted
```

filesystem on them).

"dmraid -ay -f pdc" (pdc looked up from "dmraid -l") activates all software RAID sets with Promise format discovered and ignores all other supported formats.

"dmraid -r" discovers all software RAID devices supported on your system, eg:

```
/dev/dm-46: hpt45x, "hpt45x_chidjhaiaa-0", striped, ok, 320172928 sectors, data@ 0  
/dev/dm-50: hpt45x, "hpt45x_chidjhaiaa-0", striped, ok, 320172928 sectors, data@ 0  
/dev/dm-54: hpt45x, "hpt45x_chidjhaiaa-1", striped, ok, 320172928 sectors, data@ 0  
/dev/dm-58: hpt45x, "hpt45x_chidjhaiaa-1", striped, ok, 320172928 sectors, data@ 0
```

"dmraid -s -s hpt45x_chidjhaiaa" displays properties of set "hpt45x_chidjhaiaa", eg:

*** Superset

```
name : hpt45x_chidjhaiaa  
size : 640345856  
stride : 128  
type : raid10  
status : ok  
subsets: 2  
dev : 4  
spare : 0  
---> Subset  
name : hpt45x_chidjhaiaa-0  
size : 640345856  
stride : 128  
type : stripe  
status : ok  
subsets: 0  
dev : 2  
spare : 0  
---> Subset  
name : hpt45x_chidjhaiaa-1  
size : 640345856  
stride : 128  
type : stripe  
status : ok  
subsets: 0  
dev : 2  
spare : 0
```

"dmraid -s -ccs hpt45" displays properties in column format of all sets and subsets with hpt45* format, eg:

```
hpt45x_chidjhaiaa,640345856,128,raid10,ok,4,0  
hpt45x_chidjhaiaa-a,640345856,128,stripe,ok,2,0  
hpt45x_chidjhaiaa-b,640345856,128,stripe,ok,2,0
```

"dmraid -r --sep : -cpath:size" display paths and sizes in sectors for RAID devices in

column format using ':' as a delimiter, eg:

```
/dev/dm-8:320173055  
/dev/dm-12:320173055  
/dev/dm-22:320173055  
/dev/dm-26:320173055  
/dev/dm-30:586114703  
/dev/dm-34:586114703  
/dev/dm-38:586114703  
/dev/dm-42:586114703  
/dev/dm-46:156301487  
/dev/dm-50:156301487  
/dev/dm-54:390624896  
/dev/dm-58:390624896  
/dev/dm-62:390624896  
/dev/dm-66:390624896
```

"dmraid -f isw -C Raid0 --type 0 --strip 8k --size 20g --disk "/dev/sdb /dev/sdc"" creates an ISW volume with a name of "Raid0", 20Gig bytes in total, and 8kilo bytes strip size on two disks.

"dmraid -f isw -C Test0 --type 0 --disk "/dev/sdd /dev/sde"" creates an ISW volume with the default size and strip size.

"dmraid -f isw -C Test10 --type 01 --strip 128B --disk "/dev/sda /dev/sdb /dev/sdc /dev/sdd" creates a stacked RAID device, RAID10 (isw format), with a name of "Test10", 128 blocks (512bytes) strip size , and the default volume size on 4 disks.

"dmraid -f isw -S -M /dev/sde" marks the device /dev/sde as a hot spare for rebuild

"dmraid -R isw_djaggchdde_RAID1 /dev/sde" starts rebuild of the RAID volume on device /dev/sde

Diagnostics

dmraid returns an exit code of 0 for success or 1 for error.

Author

Heinz Mauelshagen <Mauelshagen@RedHat.com>

dnf(8) — Linux manual page

NAME | SYNOPSIS | DESCRIPTION | OPTIONS | COMMANDS | SPECIFYING PACKAGES |
 SPECIFYING PROVIDES | SPECIFYING GROUPS | SPECIFYING MODULES |
 SPECIFYING TRANSACTIONS | PACKAGE FILTERING | METADATA SYNCHRONIZATION |
 CONFIGURATION FILES REPLACEMENT POLICY | FILES | SEE ALSO | AUTHOR | COPYRIGHT
 | COLOPHON

Search online pages

DNF(8)

DNF

DNF(8)

NAME top

`dnf` – DNF Command Reference

SYNOPSIS top

`dnf [options] <command> [<args>...]`

DESCRIPTION top

DNF is the next upcoming major version of **YUM**, a package manager for RPM-based Linux distributions. It roughly maintains CLI compatibility with YUM and defines a strict API for extensions and plugins.

Plugins can modify or extend features of DNF or provide additional CLI commands on top of those mentioned below. If you know the name of such a command (including commands mentioned below), you may find/install the package which provides it using the appropriate virtual provide in the form of `dnf-command(<alias>)`, where `<alias>` is the name of the command; e.g. ```dnf install 'dnf-command(versionlock)'``` installs a `versionlock` plugin. This approach also applies to specifying dependencies of packages that require a particular DNF command.

Return values:

- **0** : Operation was successful.
- **1** : An error occurred, which was handled by dnf.
- **3** : An unknown unhandled error occurred during operation.
- **100**: See `check-update`
- **200**: There was a problem with acquiring or releasing of locks.

Available commands:

- `alias`
- `autoremove`
- `check`
- `check-update`
- `clean`
- `deplist`
- `distro-sync`
- `downgrade`
- `group`
- `help`
- `history`

- *info*
- *install*
- *list*
- *makecache*
- *mark*
- *module*
- *provides*
- *reinstall*
- *remove*
- *repoinfo*
- *repolist*
- *repoquery*
- *repository-packages*
- *search*
- *shell*
- *swap*
- *updateinfo*
- *upgrade*
- *upgrade-minimal*

Additional information:

- *Options*
- *Specifying Packages*
- *Specifying Provides*
- *Specifying File Provides*
- *Specifying Groups*
- *Specifying Transactions*
- *Metadata Synchronization*
- *Configuration Files Replacement Policy*
- *Files*
- *See Also*

OPTIONS

[top](#)

- 4** Resolve to IPv4 addresses only.
- 6** Resolve to IPv6 addresses only.
- advisory=<advisory>, --advisories=<advisory>**
Include packages corresponding to the advisory ID, Eg.
FEDORA-2021-123. Applicable for the **install**, **repoquery**,
updateinfo, **upgrade** and **offline-upgrade** (`dnf-plugins-core`)
commands.
- allowerasing**
Allow erasing of installed packages to resolve
dependencies. This option could be used as an alternative
to the **yum swap** command where packages to remove are not
explicitly defined.
- assumeno**

Automatically answer no for all questions.

-b, --best

Try the best available package versions in transactions. Specifically during `dnf upgrade`, which by default skips over updates that can not be installed for dependency reasons, this switch forces DNF to only consider the latest packages. When running into packages with broken dependencies, DNF will fail giving a reason why the latest version can not be installed.

Note that the use of the newest available version is only guaranteed for the packages directly requested (e.g. as a command line arguments), and the solver may use older versions of dependencies to meet their requirements.

--bugfix

Include packages that fix a bugfix issue. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

--bz=<bugzilla>, --bzs=<bugzilla>

Include packages that fix a Bugzilla ID, Eg. 123123. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

-C, --cacheonly

Run entirely from system cache, don't update the cache and use it even in case it is expired.

DNF uses a separate cache for each user under which it executes. The cache for the root user is called the system cache. This switch allows a regular user read-only access to the system cache, which usually is more fresh than the user's and thus he does not have to wait for metadata sync.

--color=<color>

Control whether color is used in terminal output. Valid values are `always`, `never` and `auto` (default).

--comment=<comment>

Add a comment to the transaction history.

-c <config file>, --config=<config file>

Configuration file location.

--cve=<cves>, --cves=<cves>

Include packages that fix a CVE (Common Vulnerabilities and Exposures) ID (<http://cve.mitre.org/about/>), Eg. CVE-2021-0123. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

-d <debug level>, --debuglevel=<debug level>

Debugging output level. This is an integer value between 0 (no additional information strings) and 10 (shows all debugging information, even that not understandable to the user), default is 2. Deprecated, use `-v` instead.

--debugsolver

Dump data aiding in dependency solver debugging into `./debugdata`.

--disableexcludes=[all|main|<repoid>],

--disableexcludepkgs=[all|main|<repoid>]

Disable the configuration file excludes. Takes one of the following three options:

- `all`, disables all configuration file excludes
- `main`, disables excludes defined in the `[main]` section
- `repoid`, disables excludes defined for the given repository

--disable, --set-disabled

Disable specified repositories (automatically saves). The option has to be used together with the `config-manager` command (`dnf-plugins-core`).

--disableplugin=<plugin names>

Disable the listed plugins specified by names or globs.

--disablerepo=<repoid>

Temporarily disable active repositories for the purpose of the current dnf command. Accepts an id, a comma-separated list of ids, or a glob of ids. This option can be specified multiple times, but is mutually exclusive with **--repo**.

--downloaddir=<path>, --destdir=<path>

Redirect downloaded packages to provided directory. The option has to be used together with the **--downloadonly** command line option, with the **download**, **modulesync**, **reposync** or **system-upgrade** commands (dnf-plugins-core).

--downloadonly

Download the resolved package set without performing any rpm transaction (install/upgrade/erase).

Packages are removed after the next successful transaction. This applies also when used together with **--destdir** option as the directory is considered as a part of the DNF cache. To persist the packages, use the **download** command instead.

-e <error level>, --errorlevel=<error level>

Error output level. This is an integer value between 0 (no error output) and 10 (shows all error messages), default is 3. Deprecated, use **-v** instead.

--enable, --set-enabled

Enable specified repositories (automatically saves). The option has to be used together with the **config-manager** command (dnf-plugins-core).

--enableplugin=<plugin names>

Enable the listed plugins specified by names or globs.

--enablerepo=<repoid>

Temporarily enable additional repositories for the purpose of the current dnf command. Accepts an id, a comma-separated list of ids, or a glob of ids. This option can be specified multiple times.

--enhancement

Include enhancement relevant packages. Applicable for the **install**, **repoquery**, **updateinfo**, **upgrade** and **offline-upgrade** (dnf-plugins-core) commands.

-x <package-file-spec>, --exclude=<package-file-spec>

Exclude packages specified by **<package-file-spec>** from the operation.

--excludepkgs=<package-file-spec>

Deprecated option. It was replaced by the **--exclude** option.

--forcearch=<arch>

Force the use of an architecture. Any architecture can be specified. However, use of an architecture not supported natively by your CPU will require emulation of some kind. This is usually through QEMU. The behavior of **--forcearch** can be configured by using the **arch** and **ignorearch** configuration options with values **<arch>** and **True** respectively.

-h, --help, --help-cmd

Show the help.

--installroot=<path>

Specifies an alternative installroot, relative to where all packages will be installed. Think of this like doing **chroot <root> dnf**, except using **--installroot** allows dnf to work before the chroot is created. It requires absolute path.

- **cachedir**, **log files**, **releasever**, and **gpgkey** are taken from or stored in the installroot. **Gpgkeys** are imported into the installroot from a path relative to the host which can be specified in the repository section of configuration files.

- *configuration file* and *reposdir* are searched inside the installroot first. If they are not present, they are taken from the host system. Note: When a path is specified within a command line argument (`--config=<config file>` in case of *configuration file* and `--setopt=reposdir=<reposdir>` for *reposdir*) then this path is always relative to the host with no exceptions.
- *vars* are taken from the host system or installroot according to *reposdir*. When *reposdir* path is specified within a command line argument, vars are taken from the installroot. When *varsdir* paths are specified within a command line argument (`--setopt=varsdir=<reposdir>`) then those path are always relative to the host with no exceptions.
- The *pluginpath* and *pluginconfpath* are relative to the host. Note: You may also want to use the command-line option `--releasever=<release>` when creating the installroot, otherwise the *\$releasever* value is taken from the rpmdb within the installroot (and thus it is empty at the time of creation and the transaction will fail). If `--releasever=/` is used, the releasever will be detected from the host (/) system. The new installroot path at the time of creation does not contain the *repository*, *releasever* and *dnf.conf* files.

On a modular system you may also want to use the `--setopt=module_platform_id=<module_platform_name:stream>` command-line option when creating the installroot, otherwise the *module_platform_id* value will be taken from the */etc/os-release* file within the installroot (and thus it will be empty at the time of creation, the modular dependency could be unsatisfied and modules content could be excluded).

Installroot examples:

```
dnf --installroot=<installroot> --releasever=<release> install
system-release
    Permanently sets the releasever of the system in the
    <installroot> directory to <release>.

dnf --installroot=<installroot> --setopt=reposdir=<path>
--config /path/dnf.conf upgrade
    Upgrades packages inside the installroot from a
    repository described by --setopt using configuration
    from /path/dnf.conf.
```

--newpackage
Include newpackage relevant packages. Applicable for the **install**, **repoquery**, **updateinfo**, **upgrade** and **offline-upgrade** (dnf-plugins-core) commands.

--noautoremove
Disable removal of dependencies that are no longer used.
It sets *clean_requirements_on_remove* configuration option to **False**.

--nobest
Set best option to **False**, so that transactions are not limited to best candidates only.

--nodocs
Do not install documentation. Sets the rpm flag '**RPMTRANS_FLAG_NODOCS**'.

--nogpgcheck
Skip checking GPG signatures on packages (if RPM policy allows).

--nopugins
Disable all plugins.

--obsoletes
This option has an effect on an install/update, it enables dnf's obsoletes processing logic. For more information see the *obsoletes* option.

This option also displays capabilities that the package obsoletes when used together with the *repoquery* command.

Configuration Option: *obsoletes*

-q, --quiet
 In combination with a non-interactive command, shows just the relevant content. Suppresses messages notifying about the current state or actions of DNF.

-R <minutes>, --randomwait=<minutes>
 Maximum command wait time.

--refresh
 Set metadata as expired before running the command.

--releasever=<release>
 Configure DNF as if the distribution release was <release>. This can affect cache paths, values in configuration files and mirrorlist URLs.

--repofrompath <repo>,<path/url>
 Specify a repository to add to the repositories for this query. This option can be used multiple times.

- The repository label is specified by <repo>.
- The path or url to the repository is specified by <path/url>. It is the same path as a baseurl and can be also enriched by the *repo variables*.
- The configuration for the repository can be adjusted using --
`-setopt=<repo>.<option>=<value>`.
- If you want to view only packages from this repository, combine this with the `--repo=<repo>` or `--disablerepo="*"` switches.

--repo=<repoid>, --repoid=<repoid>
 Enable just specific repositories by an id or a glob. Can be used multiple times with accumulative effect. It is basically a shortcut for `--disablerepo="*"`
`--enablerepo=<repoid>` and is mutually exclusive with the `--disablerepo` option.

--rpmverbosity=<name>
 RPM debug scriptlet output level. Sets the debug level to <name> for RPM scriptlets. For available levels, see the `rpmverbosity` configuration option.

--sec-severity=<severity>, --secseverity=<severity>
 Includes packages that provide a fix for an issue of the specified severity. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

--security
 Includes packages that provide a fix for a security issue. Applicable for the `install`, `repoquery`, `updateinfo`, `upgrade` and `offline-upgrade` (`dnf-plugins-core`) commands.

--setopt=<option>=<value>
 Override a configuration option from the configuration file. To override configuration options for repositories, use `repoid.option` for the <option>. Values for configuration options like `excludepkgs`, `includepkgs`, `installonlypkgs` and `tsflags` are appended to the original value, they do not override it. However, specifying an empty value (e.g. `--setopt=tsflags=`) will clear the option.

--skip-broken
 Resolve depsolve problems by removing packages that are causing problems from the transaction. It is an alias for the `strict` configuration option with value `False`. Additionally, with the `enable` and `disable` module subcommands it allows one to perform an action even in case of broken modular dependencies.

--showduplicates
 Show duplicate packages in repositories. Applicable for the list and search commands.

-v, --verbose
 Verbose operation, show debug messages.

--version

Show DNF version and exit.

-y, --assumeyes

Automatically answer yes for all questions.

List options are comma-separated. Command-line options override respective settings from configuration files.

COMMANDS

[top](#)

For an explanation of **<package-spec>**, **<package-file-spec>** and **<package-name-spec>** see *Specifying Packages*.

For an explanation of **<provide-spec>** see *Specifying Provides*.

For an explanation of **<group-spec>** see *Specifying Groups*.

For an explanation of **<module-spec>** see *Specifying Modules*.

For an explanation of **<transaction-spec>** see *Specifying Transactions*.

Alias Command

Command: **alias**

Allows the user to define and manage a list of aliases (in the form **<name=value>**), which can be then used as dnf commands to abbreviate longer command sequences. For examples on using the alias command, see *Alias Examples*. For examples on the alias processing, see *Alias Processing Examples*.

To use an alias (**name=value**), the name must be placed as the first "command" (e.g. the first argument that is not an option). It is then replaced by its value and the resulting sequence is again searched for aliases. The alias processing stops when the first found command is not a name of any alias.

In case the processing would result in an infinite recursion, the original arguments are used instead.

Also, like in shell aliases, if the result starts with a ****, the alias processing will stop.

All aliases are defined in configuration files in the **/etc/dnf/aliases.d/** directory in the **[aliases]** section, and aliases created by the alias command are written to the **USER.conf** file. In case of conflicts, the **USER.conf** has the highest priority, and alphabetical ordering is used for the rest of the configuration files.

Optionally, there is the **enabled** option in the **[main]** section defaulting to True. This can be set for each file separately in the respective file, or globally for all aliases in the **ALIASES.conf** file.

```
dnf alias [options] [list] [<name>...]
List aliases with their final result. The [<alias>...]
parameter further limits the result to only those aliases
matching it.
```

```
dnf alias [options] add <name=value>...
Create new aliases.
```

```
dnf alias [options] delete <name>...
Delete aliases.
```

Alias Examples

dnf alias list

Lists all defined aliases.

dnf alias add rm=remove

Adds a new command alias called **rm** which works the same as the **remove** command.

**dnf alias add upgrade="\\upgrade --skip-broken
--disableexcludes-all --obsoletes"**

Adds a new command alias called **upgrade** which works the same as the **upgrade** command, with additional options. Note that the original **upgrade** command is prefixed with a **** to prevent an infinite loop in alias processing.

Alias Processing Examples

If there are defined aliases `in=install` and `FORCE="--skip-broken --disableexcludes=all"`:

- `dnf FORCE in` will be replaced with `dnf --skip-broken --disableexcludes=all install`
- `dnf in FORCE` will be replaced with `dnf install FORCE` (which will fail)

If there is defined alias `in=install`:

- `dnf in` will be replaced with `dnf install`
- `dnf --repo updates in` will be replaced with `dnf --repo updates in` (which will fail)

Autoremove Command

Command: `autoremove`

Aliases for `explicit NEVRA matching`: `autoremove-n`, `autoremove-na`, `autoremove-nevra`

dnf [options] autoremove

Removes all "leaf" packages from the system that were originally installed as dependencies of user-installed packages, but which are no longer required by any such package.

Packages listed in `installyonlypkgs` are never automatically removed by this command.

dnf [options] autoremove <spec>...

This is an alias for the `Remove Command` command with `clean_requirements_on_remove` set to `True`. It removes the specified packages from the system along with any packages depending on the packages being removed. Each `<spec>` can be either a `<package-spec>`, which specifies a package directly, or a `@<group-spec>`, which specifies an (environment) group which contains it. It also removes any dependencies that are no longer needed.

There are also a few specific autoremove commands `autoremove-n`, `autoremove-na` and `autoremove-nevra` that allow the specification of an exact argument in the NEVRA (`name-epoch:version-release.architecture`) format.

This command by default does not force a sync of expired metadata. See also `Metadata Synchronization`.

Check Command

Command: `check`

dnf [options] check [--dependencies] [--duplicates] [--obsoleted] [--provides]

Checks the local packagedb and produces information on any problems it finds. You can limit the checks to be performed by using the `--dependencies`, `--duplicates`, `--obsoleted` and `--provides` options (the default is to check everything).

Check-Update Command

Command: `check-update`
Aliases: `check-upgrade`

dnf [options] check-update [--changelogs]

[<package-file-spec>...]

Non-interactively checks if updates of the specified packages are available. If no `<package-file-spec>` is given, checks whether any updates at all are available for your system. DNF exit code will be 100 when there are updates available and a list of the updates will be printed, 0 if not and 1 if an error occurs. If `--changelogs` option is specified, also changelog delta of packages about to be updated is printed.

Please note that having a specific newer version available for an installed package (and reported by `check-update`) does not imply that subsequent `dnf upgrade` will install it. The difference is that `dnf upgrade` has restrictions (like package dependencies being satisfied) to take into account.

The output is affected by the `autocheck_running_kernel` configuration option.

Clean CommandCommand: **clean**

Performs cleanup of temporary files kept for repositories. This includes any such data left behind from disabled or removed repositories as well as for different distribution release versions.

dnf clean dbcache

Removes cache files generated from the repository metadata. This forces DNF to regenerate the cache files the next time it is run.

dnf clean expire-cache

Marks the repository metadata expired. DNF will re-validate the cache for each repository the next time it is used.

dnf clean metadata

Removes repository metadata. Those are the files which DNF uses to determine the remote availability of packages. Using this option will make DNF download all the metadata the next time it is run.

dnf clean packages

Removes any cached packages from the system.

dnf clean all

Does all of the above.

Deplist Command

```
dnf [options] deplist [<select-options>] [<query-options>]
[<package-spec>]
```

Deprecated alias for `dnf repoquery --deplist`.

Distro-Sync CommandCommand: **distro-sync**Aliases: **dsync**

Deprecated aliases: **distrosync**, **distribution-synchronization**

dnf distro-sync [<package-spec>...]

As necessary upgrades, downgrades or keeps selected installed packages to match the latest version available from any enabled repository. If no package is given, all installed packages are considered.

See also *Configuration Files Replacement Policy*.

Downgrade CommandCommand: **downgrade**Aliases: **dg****dnf [options] downgrade <package-spec>...**

Downgrades the specified packages to the highest installable package of all known lower versions if possible. When version is given and is lower than version of installed package then it downgrades to target version.

Group CommandCommand: **group**Aliases: **grp**

Deprecated aliases: **groups**, **grouplist**, **groupinstall**, **groupupdate**, **groupremove**, **grouperase**, **groupinfo**

Groups are virtual collections of packages. DNF keeps track of groups that the user selected ("marked") installed and can manipulate the comprising packages with simple commands.

dnf [options] group [summary] <group-spec>

Display overview of how many groups are installed and available. With a spec, limit the output to the matching groups. **summary** is the default groups subcommand.

dnf [options] group info <group-spec>

Display package lists of a group. Shows which packages are installed or available from a repository when **-v** is used.

dnf [options] group install [--with-optional] <group-spec>...

Mark the specified group installed and install packages it contains. Also include **optional** packages of the group if **--with-optional** is specified. All **mandatory** and **Default**

packages will be installed whenever possible. Conditional packages are installed if they meet their requirement. If the group is already (partially) installed, the command installs the missing packages from the group. Depending on the value of *obsoletes configuration option* group installation takes obsoletes into account.

dnf [options] group list <group-spec>...
 List all matching groups, either among installed or available groups. If nothing is specified, list all known groups. **--installed** and **--available** options narrow down the requested list. Records are ordered by the *display_order* tag defined in comps.xml file. Provides a list of all hidden groups by using option **--hidden**. Provides group IDs when the **-v** or **--ids** options are used.

dnf [options] group remove <group-spec>...
 Mark the group removed and remove those packages in the group from the system which do not belong to another installed group and were not installed explicitly by the user.

dnf [options] group upgrade <group-spec>...
 Upgrades the packages from the group and upgrades the group itself. The latter comprises of installing packages that were added to the group by the distribution and removing packages that got removed from the group as far as they were not installed explicitly by the user.

Groups can also be marked installed or removed without physically manipulating any packages:

dnf [options] group mark install <group-spec>...
 Mark the specified group installed. No packages will be installed by this command, but the group is then considered installed.

dnf [options] group mark remove <group-spec>...
 Mark the specified group removed. No packages will be removed by this command.

See also *Configuration Files Replacement Policy*.

Help Command

Command: **help**

dnf help [<command>]
 Displays the help text for all commands. If given a command name then only displays help for that particular command.

History Command

Command: **history**

Aliases: **hist**

The history command allows the user to view what has happened in past transactions and act according to this information (assuming the *history_record* configuration option is set).

dnf history [list] [--reverse] [<spec>...]
 The default history action is listing information about given transactions in a table. Each *<spec>* can be either a *<transaction-spec>*, which specifies a transaction directly, or a *<transaction-spec>..<transaction-spec>*, which specifies a range of transactions, or a *<package-name-spec>*, which specifies a transaction by a package which it manipulated. When no transaction is specified, list all known transactions.

--reverse
 The order of **history list** output is printed in reverse order.

dnf history info [<spec>...]
 Describe the given transactions. The meaning of *<spec>* is the same as in the *History List Command*. When no transaction is specified, describe what happened during the latest transaction.

dnf history redo <transaction-spec>|<package-file-spec>
 Repeat the specified transaction. Uses the last

transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to redo some operations due to the current state of RPMDB, it will not redo the transaction.

```
dnf history replay [--ignore-installed] [--ignore-extras]
[--skip-unavailable] <filename>
    Replay a transaction stored in file <filename> by History Store Command. The replay will perform the exact same operations on the packages as in the original transaction and will return with an error if case of any differences in installed packages or their versions. See also the Transaction JSON Format specification of the file format.
```

--ignore-installed

Don't check for the installed packages being in the same state as those recorded in the transaction.
E.g. in case there is an upgrade **foo-1.0** → **foo-2.0** stored in the transaction, but there is **foo-1.1** installed on the target system.

--ignore-extras

Don't check for extra packages pulled into the transaction on the target system. E.g. the target system may not have some dependency, which was installed on the source system. The replay errors out on this by default, as the transaction would not be the same.

--skip-unavailable

In case some packages stored in the transaction are not available on the target system, skip them instead of erroring out.

```
dnf history rollback <transaction-spec>|<package-file-spec>
    Undo all transactions performed after the specified transaction. Uses the last transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to undo some transactions due to the current state of RPMDB, it will not undo any transaction.
```

```
dnf history store [--output <output-file>] <transaction-spec>
    Store a transaction specified by <transaction-spec>. The transaction can later be replayed by the History Replay Command.
```

Warning: The stored transaction format is considered unstable and may change at any time. It will work if the same version of dnf is used to store and replay (or between versions as long as it stays the same).

```
-o <output-file>, --output=<output-file> Store the serialized transaction into <output-file>. Default is transaction.json.
```

```
dnf history undo <transaction-spec>|<package-file-spec>
    Perform the opposite operation to all operations performed in the specified transaction. Uses the last transaction (with the highest ID) if more than one transaction for given <package-file-spec> is found. If it is not possible to undo some operations due to the current state of RPMDB, it will not undo the transaction.
```

dnf history userinstalled

Show all installonly packages, packages installed outside of DNF and packages not installed as dependency. I.e. it lists packages that will stay on the system when *Autoremove Command* or *Remove Command* along with *clean_requirements_on_remove* configuration option set to True is executed. Note the same results can be accomplished with **dnf repoquery --userinstalled**, and the repoquery command is more powerful in formatting of the output.

This command by default does not force a sync of expired metadata, except for the redo, rollback, and undo subcommands. See also *Metadata Synchronization* and *Configuration Files Replacement Policy*.

Info Command

Command: **info**
 Aliases: **if**

dnf [options] info [<package-file-spec>...]
 Lists description and summary information about installed
 and available packages.

The **info** command limits the displayed packages the same way as
 the *list command*.

This command by default does not force a sync of expired
 metadata. See also *Metadata Synchronization*.

Install Command

Command: **install**

Aliases: **in**

Aliases for *explicit NEVRA matching*: **install-n**, **install-na**, **install-nevra**
 Deprecated aliases: **localinstall**

dnf [options] install <spec>...

Makes sure that the given packages and their dependencies
 are installed on the system. Each **<spec>** can be either a
<package-spec>, or a **@<module-spec>**, or a **@<group-spec>**.
 See *Install Examples*. If a given package or provide
 cannot be (and is not already) installed, the exit code
 will be non-zero. If the **<spec>** matches both a **@-**
<module-spec> and a **@<group-spec>**, only the module is
 installed.

When **<package-spec>** to specify the exact version of the
 package is given, DNF will install the desired version, no
 matter which version of the package is already installed.
 The former version of the package will be removed in the
 case of non-installonly package.

On the other hand if **<package-spec>** specifies only a name,
 DNF also takes into account packages obsoleting it when
 picking which package to install. This behaviour is
 specific to the install command. Note that this can lead
 to seemingly unexpected results if a package has multiple
 versions and some older version is being obsoleted. It
 creates a split in the upgrade-path and both ways are
 considered correct, the resulting package is picked simply
 by lexicographical order.

There are also a few specific install commands **install-n**,
install-na and **install-nevra** that allow the specification
 of an exact argument in the NEVRA format.

See also *Configuration Files Replacement Policy*.

Install Examples

dnf install tito

Install the **tito** package (tito is the package name).

dnf install ~/Downloads/tito-0.6.2-1.fc22.noarch.rpm

Install a local rpm file tito-0.6.2-1.fc22.noarch.rpm from
 the **~/Downloads/** directory.

dnf install tito-0.5.6-1.fc22

Install the package with a specific version. If the
 package is already installed it will automatically try to
 downgrade or upgrade to the specific version.

dnf --best install tito

Install the latest available version of the package. If
 the package is already installed it will try to
 automatically upgrade to the latest version. If the latest
 version of the package cannot be installed, the
 installation will fail.

dnf install vim

DNF will automatically recognize that vim is not a package
 name, but will look up and install a package that provides
 vim with all the required dependencies. Note: Package name
 match has precedence over package provides match.

dnf install

<https://kojipkgs.fedoraproject.org//packages/tito/0.6.0/1.fc22/noarch/tito-0.6.0-1.fc22.noarch.rpm>
 Install a package directly from a URL.

```
dnf install '@docker'
    Install all default profiles of module 'docker' and their
    RPMs. Module streams get enabled accordingly.

dnf install '@Web Server'
    Install the 'Web Server' environmental group.

dnf install /usr/bin/rpmsign
    Install a package that provides the /usr/bin/rpmsign file.

dnf -y install tito --setopt=install_weak_deps=False
    Install the tito package (tito is the package name)
    without weak deps. Weak deps are not required for core
    functionality of the package, but they enhance the
    original package (like extended documentation, plugins,
    additional functions, etc.).

dnf install --advisory=FEDORA-2018-b7b99fe852 \*
    Install all packages that belong to the
    "FEDORA-2018-b7b99fe852" advisory.
```

List Command

Command: **list**
 Aliases: **ls**

Prints lists of packages depending on the packages' relation to the system. A package is **installed** if it is present in the RPMDB, and it is **available** if it is not installed but is present in a repository that DNF knows about.

The list command also limits the displayed packages according to specific criteria, e.g. to only those that update an installed package (respecting the repository **priority**). The **exclude** option in the configuration file can influence the result, but if the **--disableexcludes** command line option is used, it ensures that all installed packages will be listed.

```
dnf [options] list [--all] [<package-file-spec>...]
    Lists all packages, present in the RPMDB, in a repository
    or both.

dnf [options] list --installed [<package-file-spec>...]
    Lists installed packages.

dnf [options] list --available [<package-file-spec>...]
    Lists available packages.

dnf [options] list --extras [<package-file-spec>...]
    Lists extras, that is packages installed on the system
    that are not available in any known repository.

dnf [options] list --obsoletes [<package-file-spec>...]
    List packages installed on the system that are obsoleted
    by packages in any known repository.

dnf [options] list --recent [<package-file-spec>...]
    List packages recently added into the repositories.

dnf [options] list --upgrades [<package-file-spec>...]
    List upgrades available for the installed packages.

dnf [options] list --autoremove
    List packages which will be removed by the dnf autoremove
    command.
```

This command by default does not force a sync of expired metadata. See also **Metadata Synchronization**.

Makecache Command

Command: **makecache**
 Aliases: **mc**

```
dnf [options] makecache
    Downloads and caches metadata for enabled repositories.
    Tries to avoid downloading whenever possible (e.g. when
    the local metadata hasn't expired yet or when the metadata
    timestamp hasn't changed).

dnf [options] makecache --timer
    Like plain makecache, but instructs DNF to be more
    resource-aware, meaning it will not do anything if running
```

on battery power and will terminate immediately if it's too soon after the last successful `makecache` run (see `dnf.conf(5)`, `metadata_timer_sync`).

Mark Command

Command: `mark`

dnf mark install <package-spec>...

Marks the specified packages as installed by user. This can be useful if any package was installed as a dependency and is desired to stay on the system when `Autoremove Command` or `Remove Command` along with `clean_requirements_on_remove` configuration option set to `True` is executed.

dnf mark remove <package-spec>...

Unmarks the specified packages as installed by user. Whenever you as a user don't need a specific package you can mark it for removal. The package stays installed on the system but will be removed when `Autoremove Command` or `Remove Command` along with `clean_requirements_on_remove` configuration option set to `True` is executed. You should use this operation instead of `Remove Command` if you're not sure whether the package is a requirement of other user installed packages on the system.

dnf mark group <package-spec>...

Marks the specified packages as installed by group. This can be useful if any package was installed as a dependency or a user and is desired to be protected and handled as a group member like during group remove.

Module Command

Command: `module`

Modularity overview is available at [man page `dnf.modularity\(7\)`](#). Module subcommands take `<module-spec>...` arguments that specify modules or profiles.

dnf [options] module install <module-spec>...

Install module profiles, including their packages. In case no profile was provided, all default profiles get installed. Module streams get enabled accordingly.

This command cannot be used for switching module streams. Use the `dnf module switch-to` command for that.

dnf [options] module update <module-spec>...

Update packages associated with an active module stream, optionally restricted to a profile. If the `profile_name` is provided, only the packages referenced by that profile will be updated.

dnf [options] module switch-to <module-spec>...

Switch to or enable a module stream, change versions of installed packages to versions provided by the new stream, and remove packages from the old stream that are no longer available. It also updates installed profiles if they are available for the new stream. When a profile was provided, it installs that profile and does not update any already installed profiles.

This command can be used as a stronger version of the `dnf module enable` command, which not only enables modules, but also does a `distroSync` to all modular packages in the enabled modules.

It can also be used as a stronger version of the `dnf module install` command, but it requires to specify profiles that are supposed to be installed, because `switch-to` command does not use `default profiles`. The `switch-to` command doesn't only install profiles, it also makes a `distroSync` to all modular packages in the installed module.

dnf [options] module remove <module-spec>...

Remove installed module profiles, including packages that were installed with the `dnf module install` command. Will not remove packages required by other installed module profiles or by other user-installed packages. In case no profile was provided, all installed profiles get removed.

```
dnf [options] module remove --all <module-spec>...
    Remove installed module profiles, including packages that
    were installed with the dnf module install command. With
    --all option it additionally removes all packages whose
    names are provided by specified modules. Packages required
    by other installed module profiles and packages whose
    names are also provided by any other module are not
    removed.

dnf [options] module enable <module-spec>...
    Enable a module stream and make the stream RPMs available
    in the package set.

    Modular dependencies are resolved, dependencies checked
    and also recursively enabled. In case of modular
    dependency issue the operation will be rejected. To
    perform the action anyway please use --skip-broken option.

    This command cannot be used for switching module streams.
    Use the dnf module switch-to command for that.

dnf [options] module disable <module-name>...
    Disable a module. All related module streams will become
    unavailable. Consequently, all installed profiles will be
    removed and the module RPMs will become unavailable in the
    package set. In case of modular dependency issue the
    operation will be rejected. To perform the action anyway
    please use --skip-broken option.

dnf [options] module reset <module-name>...
    Reset module state so it's no longer enabled or disabled.
    Consequently, all installed profiles will be removed and
    only RPMs from the default stream will be available in the
    package set.

dnf [options] module provides <package-name-spec>...
    Lists all modular packages matching <package-name-spec>
    from all modules (including disabled), along with the
    modules and streams they belong to.

dnf [options] module list [--all] [module_name...]
    Lists all module streams, their profiles and states
    (enabled, disabled, default).

dnf [options] module list --enabled [module_name...]
    Lists module streams that are enabled.

dnf [options] module list --disabled [module_name...]
    Lists module streams that are disabled.

dnf [options] module list --installed [module_name...]
    List module streams with installed profiles.

dnf [options] module info <module-spec>...
    Print detailed information about given module stream.

dnf [options] module info --profile <module-spec>...
    Print detailed information about given module profiles.

dnf [options] module repoquery <module-spec>...
    List all available packages belonging to selected modules.

dnf [options] module repoquery --available <module-spec>...
    List all available packages belonging to selected modules.

dnf [options] module repoquery --installed <module-spec>...
    List all installed packages with same name like packages
    belonging to selected modules.
```

Provides Command

Command: **provides**
 Aliases: **prov**, **whatprovides**, **wp**

```
dnf [options] provides <provide-spec>
    Finds the packages providing the given <provide-spec>.
    This is useful when one knows a filename and wants to find
    what package (installed or not) provides this file. The
    <provide-spec> is gradually looked for at following
    locations:
```

1. The **<provide-spec>** is matched with all file provides of any available package:

```
$ dnf provides /usr/bin/gzip
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
Matched from:
Filename      : /usr/bin/gzip
```

2. Then all provides of all available packages are searched:

```
$ dnf provides "gzip(x86-64)"
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
Matched from:
Provide      : gzip(x86-64) = 1.9-9.fc29
```

3. DNF assumes that the **<provide-spec>** is a system command, prepends it with **/usr/bin/**, **/usr/sbin/** prefixes (one at a time) and does the file provides search again. For legacy reasons (packages that didn't do `UsrMove`) also **/bin** and **/sbin** prefixes are being searched:

```
$ dnf provides zless
gzip-1.9-9.fc29.x86_64 : The GNU data compression program
Matched from:
Filename      : /usr/bin/zless
```

4. If this last step also fails, DNF returns "Error: No Matches found".

This command by default does not force a sync of expired metadata. See also [Metadata Synchronization](#).

Reinstall Command

Command: **reinstall**
Aliases: **rei**

```
dnf [options] reinstall <package-spec>...
Installs the specified packages, fails if some of the
packages are either not installed or not available (i.e.
there is no repository where to download the same RPM).
```

Remove Command

Command: **remove**
Aliases: **rm**
Aliases for [explicit NEVRA matching](#): **remove-n**, **remove-na**, **remove-nevra**
Deprecated aliases: **erase**, **erase-n**, **erase-na**, **erase-nevra**

```
dnf [options] remove <package-spec>...
Removes the specified packages from the system along with
any packages depending on the packages being removed. Each
<spec> can be either a <package-spec>, which specifies a
package directly, or a @<group-spec>, which specifies an
(environment) group which contains it. If
clean_requirements_on_remove is enabled (the default),
also removes any dependencies that are no longer needed.
```

```
dnf [options] remove --duplicates
Removes older versions of duplicate packages. To ensure
the integrity of the system it reinstalls the newest
package. In some cases the command cannot resolve
conflicts. In such cases the dnf shell command with remove
--duplicates and upgrade dnf-shell sub-commands could
help.
```

```
dnf [options] remove --oldinstallonly
Removes old installonly packages, keeping only latest
versions and version of running kernel.
```

There are also a few specific remove commands **remove-n**, **remove-na** and **remove-nevra** that allow the specification of an exact argument in the NEVRA format.

Remove Examples

```
dnf remove acpi tito
Remove the acpi and tito packages.
```

```
dnf remove $(dnf repoquery --extras --exclude=tito,acpi)
Remove packages not present in any repository, but don't
remove the tito and acpi packages (they still might be
```

removed if they depend on some of the removed packages).

Remove older versions of duplicated packages (an equivalent of yum's **package-cleanup --cleandups**):

```
dnf remove --duplicates
```

Repoinfo Command

Command: **repoinfo**

An alias for the **repolist** command that provides more detailed information like **dnf repolist -v**.

Repolist Command

Command: **repolist**

```
dnf [options] repolist [--enabled|--disabled|--all]
```

Depending on the exact command lists enabled, disabled or all known repositories. Lists all enabled repositories by default. Provides more detailed information when **-v** option is used.

This command by default does not force a sync of expired metadata. See also [Metadata Synchronization](#).

Repoquery Command

Command: **repoquery**

Aliases: **rq**

Aliases for [explicit NEVRA matching](#): **repoquery-n**, **repoquery-na**, **repoquery-nevra**

```
dnf [options] repoquery [<select-options>] [<query-options>]  
[<package-file-spec>]
```

Searches available DNF repositories for selected packages and displays the requested information about them. It is an equivalent of **rpm -q** for remote repositories.

```
dnf [options] repoquery --groupmember <package-spec>...  
List groups that contain <package-spec>.
```

```
dnf [options] repoquery --querytags
```

Provides the list of tags recognized by the [--queryformat](#) **repoquery** option.

There are also a few specific repoquery commands **repoquery-n**, **repoquery-na** and **repoquery-nevra** that allow the specification of an exact argument in the NEVRA format (does not affect arguments of options like **--whatprovides <arg>**, ...).

Select Options

Together with **<package-file-spec>**, control what packages are displayed in the output. If **<package-file-spec>** is given, limits the resulting set of packages to those matching the specification. All packages are considered if no **<package-file-spec>** is specified.

<package-file-spec>

Package specification in the NEVRA format (name[-epoch:]version[-release]][.arch]), a package provide or a file provide. See [Specifying Packages](#).

-a, --all

Query all packages (for rpmquery compatibility, also a shorthand for **repoquery '*' or repoquery without arguments**).

--arch <arch>[,<arch>...], --archlist <arch>[,<arch>...]

Limit the resulting set only to packages of selected architectures (default is all architectures). In some cases the result is affected by the basearch of the running system, therefore to run **repoquery** for an arch incompatible with your system use the **--forcearch=<arch>** option to change the basearch.

--duplicates

Limit the resulting set to installed duplicate packages (i.e. more package versions for the same name and architecture). Installonly packages are excluded from this set.

--unneeded

Limit the resulting set to leaves packages that were installed as dependencies so they are no longer needed. This switch lists packages that are going to be removed after executing the **dnf autoremove** command.

--available

Limit the resulting set to available packages only (set by default).

--disable-modular-filtering

Disables filtering of modular packages, so that packages of inactive module streams are included in the result.

--extras

Limit the resulting set to packages that are not present in any of the available repositories.

-f <file>, --file <file>

Limit the resulting set only to the package that owns <file>.

--installed

Limit the resulting set to installed packages only. The **exclude** option in the configuration file might influence the result, but if the command line option **--disableexcludes** is used, it ensures that all installed packages will be listed.

--installonly

Limit the resulting set to installed installonly packages.

--latest-limit <number>

Limit the resulting set to <number> of latest packages for every package name and architecture. If <number> is negative, skip <number> of latest packages. For a negative <number> use the **--latest-limit=<number>** syntax.

--recent

Limit the resulting set to packages that were recently edited.

--repo <repoid>

Limit the resulting set only to packages from a repository identified by <repoid>. Can be used multiple times with accumulative effect.

--unsatisfied

Report unsatisfied dependencies among installed packages (i.e. missing requires and existing conflicts).

--upgrades

Limit the resulting set to packages that provide an upgrade for some already installed package.

--userinstalled

Limit the resulting set to packages installed by the user. The **exclude** option in the configuration file might influence the result, but if the command line option **--disableexcludes** is used, it ensures that all installed packages will be listed.

--whatdepends <capability>[,<capability>...]

Limit the resulting set only to packages that require, enhance, recommend, suggest or supplement any of <capabilities>.

--whatconflicts <capability>[,<capability>...]

Limit the resulting set only to packages that conflict with any of <capabilities>.

--whatenhances <capability>[,<capability>...]

Limit the resulting set only to packages that enhance any of <capabilities>. Use **--whatdepends** if you want to list all depending packages.

--whatobsoletes <capability>[,<capability>...]

Limit the resulting set only to packages that obsolete any of <capabilities>.

--whatprovides <capability>[,<capability>...]

Limit the resulting set only to packages that provide any

```

of <capabilities>.

--whatrecommends <capability>[,<capability>...]
    Limit the resulting set only to packages that recommend
    any of <capabilities>. Use --whatdepends if you want to
    list all depending packages.

--whatrequires <capability>[,<capability>...]
    Limit the resulting set only to packages that require any
    of <capabilities>. Use --whatdepends if you want to list
    all depending packages.

--whatsuggests <capability>[,<capability>...]
    Limit the resulting set only to packages that suggest any
    of <capabilities>. Use --whatdepends if you want to list
    all depending packages.

--whatsupplements <capability>[,<capability>...]
    Limit the resulting set only to packages that supplement
    any of <capabilities>. Use --whatdepends if you want to
    list all depending packages.

--alldeps
    This option is stackable with --whatrequires or --
    --whatdepends only. Additionally it adds all packages
    requiring the package features to the result set (used as
    default).

--exactdeps
    This option is stackable with --whatrequires or --
    --whatdepends only. Limit the resulting set only to
    packages that require <capability> specified by
    --whatrequires.

--srpm Operate on the corresponding source RPM.

Query Options
Set what information is displayed about each package.

The following are mutually exclusive, i.e. at most one can be
specified. If no query option is given, matching packages are
displayed in the standard NEVRA notation.

-i, --info
    Show detailed information about the package.

-l, --list
    Show the list of files in the package.

-s, --source
    Show the package source RPM name.

--changelogs
    Print the package changelogs.

--conflicts
    Display capabilities that the package conflicts with. Same
    as --qf "%{conflicts}".

--depends
    Display capabilities that the package depends on,
    enhances, recommends, suggests or supplements.

--enhances
    Display capabilities enhanced by the package. Same as --qf
    "%{enhances}".

--location
    Show a location where the package could be downloaded
    from.

--obsoletes
    Display capabilities that the package obsoletes. Same as
    --qf "%{obsoletes}".

--provides
    Display capabilities provided by the package. Same as --qf
    "%{provides}".

--recommends
    Display capabilities recommended by the package. Same as

```

```
--qf "%{recommends}".

--requires
    Display capabilities that the package depends on. Same as
    --qf "%{requires}".

--requires-pre
    Display capabilities that the package depends on for
    running a %pre script. Same as --qf "%{requires-pre}".

--suggests
    Display capabilities suggested by the package. Same as
    --qf "%{suggests}".

--supplements
    Display capabilities supplemented by the package. Same as
    --qf "%{supplements}".

--tree Display a recursive tree of packages with capabilities
        specified by one of the following supplementary options:
        --whatrequires, --requires, --conflicts, --enhances,
        --suggests, --provides, --supplements, --recommends.

--deplist
    Produce a list of all direct dependencies and what
    packages provide those dependencies for the given
    packages. The result only shows the newest providers
    (which can be changed by using --verbose).

--nvr Show found packages in the name-version-release format.
    Same as --qf "%{name}-%{version}-%{release}".

--nevra
    Show found packages in the
    name-epoch:version-release.architecture format. Same as
    --qf "%{name}-%{epoch}: %{version}-%{release}.%{arch}"
    (default).

--envra
    Show found packages in the
    epoch:name-version-release.architecture format. Same as
    --qf "%{epoch}: %{name}-%{version}-%{release}.%{arch}"

--qf <format>, --queryformat <format>
    Custom display format. <format> is the string to output
    for each matched package. Every occurrence of %{<tag>}
    within is replaced by the corresponding attribute of the
    package. The list of recognized tags can be displayed by
    running dnf repoquery --querytags.

--recursive
    Query packages recursively. Has to be used with
    --whatrequires <REQ> (optionally with --alldeps, but not
    with --exactdeps) or with --requires <REQ> --resolve.

--resolve
    resolve capabilities to originating package(s).
```

Examples

Display NEVRAs of all available packages matching **light***:

```
dnf repoquery 'light*'
```

Display NEVRAs of all available packages matching name **light*** and
 architecture **noarch** (accepts only arguments in the
 "<name>.arch" format):

```
dnf repoquery-na 'light*.noarch'
```

Display requires of all lighttpd packages:

```
dnf repoquery --requires lighttpd
```

Display packages providing the requires of python packages:

```
dnf repoquery --requires python --resolve
```

Display source rpm of lighttpd package:

```
dnf repoquery --source lighttpd
```

Display package name that owns the given file:

```
dnf repoquery --file /etc/lighttpd/lighttpd.conf
```

Display name, architecture and the containing repository of all lighttpd packages:

```
dnf repoquery --queryformat '%{name}.%{arch} : %{reponame}' lighttpd
```

Display all available packages providing "webserver":

```
dnf repoquery --whatprovides webserver
```

Display all available packages providing "webserver" but only for "i686" architecture:

```
dnf repoquery --whatprovides webserver --arch i686
```

Display duplicate packages:

```
dnf repoquery --duplicates
```

Display source packages that require a <provide> for a build:

```
dnf repoquery --disablerepo="*" --enablerepo="*-source" --arch=src --whatrequires <provide>
```

Repository-Packages Command

Command: **repository-packages**

Deprecated aliases: **repo-pkgs**, **repo-packages**, **repository-pkgs**

The repository-packages command allows the user to run commands on top of all packages in the repository named <repoid>. However, any dependency resolution takes into account packages from all enabled repositories. The <package-file-spec> and <package-spec> specifications further limit the candidates to only those packages matching at least one of them.

The **info** subcommand lists description and summary information about packages depending on the packages' relation to the repository. The **list** subcommand just prints lists of those packages.

```
dnf [options] repository-packages <repoid> check-update
[<package-file-spec>...]
    Non-interactively checks if updates of the specified
    packages in the repository are available. DNF exit code
    will be 100 when there are updates available and a list of
    the updates will be printed.
```

```
dnf [options] repository-packages <repoid> info [--all]
[<package-file-spec>...]
    List all related packages.
```

```
dnf [options] repository-packages <repoid> info --installed
[<package-file-spec>...]
    List packages installed from the repository.
```

```
dnf [options] repository-packages <repoid> info --available
[<package-file-spec>...]
    List packages available in the repository but not
    currently installed on the system.
```

```
dnf [options] repository-packages <repoid> info --extras
[<package-file-specs>...]
    List packages installed from the repository that are not
    available in any repository.
```

```
dnf [options] repository-packages <repoid> info --obsoletes
[<package-file-spec>...]
    List packages in the repository that obsolete packages
    installed on the system.
```

```
dnf [options] repository-packages <repoid> info --recent
[<package-file-spec>...]
    List packages recently added into the repository.
```

```
dnf [options] repository-packages <repoid> info --upgrades
[<package-file-spec>...]
    List packages in the repository that upgrade packages
    installed on the system.
```

```

dnf [options] repository-packages <repoid> install
[<package-spec>...]
    Install packages matching <package-spec> from the
    repository. If <package-spec> isn't specified at all,
    install all packages from the repository.

dnf [options] repository-packages <repoid> list [--all]
[<package-file-spec>...]
    List all related packages.

dnf [options] repository-packages <repoid> list --installed
[<package-file-spec>...]
    List packages installed from the repository.

dnf [options] repository-packages <repoid> list --available
[<package-file-spec>...]
    List packages available in the repository but not
    currently installed on the system.

dnf [options] repository-packages <repoid> list --extras
[<package-file-spec>...]
    List packages installed from the repository that are not
    available in any repository.

dnf [options] repository-packages <repoid> list --obsoletes
[<package-file-spec>...]
    List packages in the repository that obsolete packages
    installed on the system.

dnf [options] repository-packages <repoid> list --recent
[<package-file-spec>...]
    List packages recently added into the repository.

dnf [options] repository-packages <repoid> list --upgrades
[<package-file-spec>...]
    List packages in the repository that upgrade packages
    installed on the system.

dnf [options] repository-packages <repoid> move-to
[<package-spec>...]
    Reinstall all those packages that are available in the
    repository.

dnf [options] repository-packages <repoid> reinstall
[<package-spec>...]
    Run the reinstall-old subcommand. If it fails, run the
    move-to subcommand.

dnf [options] repository-packages <repoid> reinstall-old
[<package-spec>...]
    Reinstall all those packages that were installed from the
    repository and simultaneously are available in the
    repository.

dnf [options] repository-packages <repoid> remove
[<package-spec>...]
    Remove all packages installed from the repository along
    with any packages depending on the packages being removed.
    If clean_requirements_on_remove is enabled (the default)
    also removes any dependencies that are no longer needed.

dnf [options] repository-packages <repoid> remove-or-distro-sync
[<package-spec>...]
    Select all packages installed from the repository.
    Upgrade, downgrade or keep those of them that are
    available in another repository to match the latest
    version available there and remove the others along with
    any packages depending on the packages being removed. If
    clean_requirements_on_remove is enabled (the default) also
    removes any dependencies that are no longer needed.

dnf [options] repository-packages <repoid> remove-or-reinstall
[<package-spec>...]
    Select all packages installed from the repository.
    Reinstall those of them that are available in another
    repository and remove the others along with any packages
    depending on the packages being removed. If
    clean_requirements_on_remove is enabled (the default) also
    removes any dependencies that are no longer needed.

dnf [options] repository-packages <repoid> upgrade

```

[<package-spec>...]
 Update all packages to the highest resolvable version available in the repository. When versions are specified in the **<package-spec>**, update to these versions.

dnf [options] repository-packages <repoid> upgrade-to [<package-specs>...]
 A deprecated alias for the upgrade subcommand.

Search Command

Command: **search**
 Aliases: **se**

dnf [options] search [--all] <keywords>...
 Search package metadata for keywords. Keywords are matched as case-insensitive substrings, globbing is supported. By default lists packages that match all requested keys (AND operation). Keys are searched in package names and summaries. If the "--all" option is used, lists packages that match at least one of the keys (an OR operation). In addition the keys are searched in the package descriptions and URLs. The result is sorted from the most relevant results to the least.

This command by default does not force a sync of expired metadata. See also *Metadata Synchronization*.

Shell Command

Command: **shell**
 Aliases: **sh**

dnf [options] shell [filename]
 Open an interactive shell for conducting multiple commands during a single execution of DNF. These commands can be issued manually or passed to DNF from a file. The commands are much the same as the normal DNF command line options. There are a few additional commands documented below.

config [conf-option] [value]

- Set a configuration option to a requested value.
 If no value is given it prints the current value.

repo [list|enable|disable] [repo-id]

- list: list repositories and their status
- enable: enable repository
- disable: disable repository

transaction [list|reset|solve|run]

- list: resolve and list the content of the transaction
- reset: reset the transaction
- run: resolve and run the transaction

Note that all local packages must be used in the first shell transaction subcommand (e.g. **install /tmp/nodejs-1.1.x86_64.rpm /tmp/acpi-1-1.noarch.rpm**) otherwise an error will occur. Any **disable**, **enable**, and **reset** module operations (e.g. **module enable nodejs**) must also be performed before any other shell transaction subcommand is used.

Swap Command

Command: **swap**

dnf [options] swap <remove-spec> <install-spec>
 Remove spec and install spec in one transaction. Each **<spec>** can be either a **<package-spec>**, which specifies a package directly, or a **@<group-spec>**, which specifies an (environment) group which contains it. Automatic conflict solving is provided in DNF by the --allowerasing option that provides the functionality of the swap command automatically.

Updateinfo Command

Command: **updateinfo**

```

Aliases: upif
Deprecated aliases: list-updateinfo, list-security, list-sec, info-updateinfo, info-security, info-sec, summ

dnf [options] updateinfo [--summary|--list|--info]
[<availability>] [<spec>...]
    Display information about update advisories.

Depending on the output type, DNF displays just counts of
advisory types (omitted or --summary), list of advisories
(--list) or detailed information (--info). The -v option
extends the output. When used with --info, the information
is even more detailed. When used with --list, an
additional column with date of the last advisory update is
added.

<availability> specifies whether advisories about newer
versions of installed packages (omitted or --available),
advisories about equal and older versions of installed
packages (--installed), advisories about newer versions of
those installed packages for which a newer version is
available (--updates) or advisories about any versions of
installed packages (--all) are taken into account. Most of
the time, --available and --updates displays the same
output. The outputs differ only in the cases when an
advisory refers to a newer version but there is no enabled
repository which contains any newer version.

Note, that --available takes only the latest installed
versions of packages into account. In case of the kernel
packages (when multiple version could be installed
simultaneously) also packages of the currently running
version of kernel are added.

To print only advisories referencing a CVE or a bugzilla
use --with-cve or --with-bz options. When these switches
are used also the output of the --list is altered - the ID
of the CVE or the bugzilla is printed instead of the one
of the advisory.

If given and if neither ID, type (bugfix, enhancement,
security/sec) nor a package name of an advisory matches
<spec>, the advisory is not taken into account. The
matching is case-sensitive and in the case of advisory IDs
and package names, globbing is supported.

Output of the --summary option is affected by the
autocheck\_running\_kernel configuration option.

```

Upgrade Command

```

Command: upgrade
Aliases: up
Deprecated aliases: update, upgrade-to, update-to, localupdate

```

```

dnf [options] upgrade
    Updates each package to the latest version that is both
    available and resolvable.

```

```

dnf [options] upgrade <package-spec>...
    Updates each specified package to the latest available
    version. Updates dependencies as necessary. When versions
    are specified in the <package-spec>, update to these
    versions.

```

```

dnf [options] upgrade @<spec>...
    Alias for the dnf module update command.

```

If the main **obsoletes** configure option is true or the **--obsoletes** flag is present, dnf will include package obsoletes in its calculations. For more information see **obsoletes**.

See also [Configuration Files Replacement Policy](#).

Upgrade-Minimal Command

```

Command: upgrade-minimal
Aliases: up-min
Deprecated aliases: update-minimal

```

```

dnf [options] upgrade-minimal
    Updates each package to the latest available version that
    provides a bugfix, enhancement or a fix for a security
    issue (security).

```

```
dnf [options] upgrade-minimal <package-spec>...
    Updates each specified package to the latest available
    version that provides a bugfix, enhancement or a fix for
    security issue (security). Updates dependencies as
    necessary.
```

SPECIFYING PACKAGES [top](#)

Many commands take a **<package-spec>** parameter that selects a package for the operation. The **<package-spec>** argument is matched against package NEVRAs, provides and file provides.

<package-file-spec> is similar to **<package-spec>**, except provides matching is not performed. Therefore, **<package-file-spec>** is matched only against NEVRAs and file provides.

<package-name-spec> is matched against NEVRAs only.

Globs

Package specification supports the same glob pattern matching that shell does, in all three above mentioned packages it matches against (NEVRAs, provides and file provides).

The following patterns are supported:

- * Matches any number of characters.
- ? Matches any single character.
- [] Matches any one of the enclosed characters. A pair of characters separated by a hyphen denotes a range expression; any character that falls between those two characters, inclusive, is matched. If the first character following the [is a ! or a ^ then any character not enclosed is matched.

Note: Curly brackets ({}) are not supported. You can still use them in shells that support them and let the shell do the expansion, but if quoted or escaped, dnf will not expand them.

NEVRA Matching

When matching against NEVRAs, partial matching is supported. DNF tries to match the spec against the following list of NEVRA forms (in decreasing order of priority):

- **name-[epoch:]version-release.arch**
- **name.arch**
- **name**
- **name-[epoch:]version-release**
- **name-[epoch:]version**

Note that **name** can in general contain dashes (e.g. **package-with-dashes**).

The first form that matches any packages is used and the remaining forms are not tried. If none of the forms match any packages, an attempt is made to match the **<package-spec>** against full package NEVRAs. This is only relevant if globs are present in the **<package-spec>**.

<package-spec> matches NEVRAs the same way **<package-name-spec>** does, but in case matching NEVRAs fails, it attempts to match against provides and file provides of packages as well.

You can specify globs as part of any of the five NEVRA components. You can also specify a glob pattern to match over multiple NEVRA components (in other words, to match across the NEVRA separators). In that case, however, you need to write the spec to match against full package NEVRAs, as it is not possible to split such spec into NEVRA forms.

Specifying NEVRA Matching Explicitly

Some commands (**autoremove**, **install**, **remove** and **repoquery**) also have aliases with suffixes **-n**, **-na** and **-nevra** that allow to explicitly specify how to parse the arguments:

- Command `install-n` only matches against `name`.
- Command `install-na` only matches against `name.arch`.
- Command `install-nevra` only matches against
`name-[epoch:]version-release.arch`.

SPECIFYING PROVIDES[top](#)

`<provide-spec>` in command descriptions means the command operates on packages providing the given spec. This can either be an explicit provide, an implicit provide (i.e. name of the package) or a file provide. The selection is case-sensitive and globbing is supported.

Specifying File Provides

If a spec starts with either `/` or `*/`, it is considered as a potential file provide.

SPECIFYING GROUPS[top](#)

`<group-spec>` allows one to select (environment) groups a particular operation should work on. It is a case insensitive string (supporting globbing characters) that is matched against a group's ID, canonical name and name translated into the current `LC_MESSAGES` locale (if possible).

SPECIFYING MODULES[top](#)

`<module-spec>` allows one to select modules or profiles a particular operation should work on.

It is in the form of `NAME:STREAM:VERSION:CONTEXT:ARCH/PROFILE` and supported partial forms are the following:

- `NAME`
- `NAME:STREAM`
- `NAME:STREAM:VERSION`
- `NAME:STREAM:VERSION:CONTEXT`
- all above combinations with `::ARCH` (e.g. `NAME::ARCH`)
- `NAME:STREAM:VERSION:CONTEXT:ARCH`
- all above combinations with `/PROFILE` (e.g. `NAME/PROFILE`)

In case stream is not specified, the enabled or the default stream is used, in this order. In case profile is not specified, the system default profile or the 'default' profile is used.

SPECIFYING TRANSACTIONS[top](#)

`<transaction-spec>` can be in one of several forms. If it is an integer, it specifies a transaction ID. Specifying `last` is the same as specifying the ID of the most recent transaction. The last form is `last-<offset>`, where `<offset>` is a positive integer. It specifies offset-th transaction preceding the most recent transaction.

PACKAGE FILTERING[top](#)

Package filtering filters packages out from the available package set, making them invisible to most of dnf commands. They cannot be used in a transaction. Packages can be filtered out by either Exclude Filtering or Modular Filtering.

Exclude Filtering

Exclude Filtering is a mechanism used by a user or by a DNF plugin to modify the set of available packages. Exclude Filtering can be modified by either `includepkgs` or `excludepkgs` configuration options in `configuration files`. The `--disableexcludes` command line option can be used to override

excludes from configuration files. In addition to user-configured excludes, plugins can also extend the set of excluded packages. To disable excludes from a DNF plugin you can use the `--disableplugin` command line option.

To disable all excludes for e.g. the `install` command you can use the following combination of command line options:

```
dnf --disableexcludes=all --disableplugin="*" install bash
```

Modular Filtering

Please see [the modularity documentation](#) for details on how Modular Filtering works.

With modularity, only RPM packages from **active** module streams are included in the available package set. RPM packages from **inactive** module streams, as well as non-modular packages with the same name or provides as a package from an **active** module stream, are filtered out. Modular filtering is not applied to packages added from the command line, installed packages, or packages from repositories with `module_hotfixes=true` in their `.repo` file.

Disabling of modular filtering is not recommended, because it could cause the system to get into a broken state. To disable modular filtering for a particular repository, specify `module_hotfixes=true` in the `.repo` file or use `--setopt=<repo_id>.module_hotfixes=true`.

To discover the module which contains an excluded package use `dnf module provides`.

METADATA SYNCHRONIZATION

[top](#)

Correct operation of DNF depends on having access to up-to-date data from all enabled repositories but contacting remote mirrors on every operation considerably slows it down and costs bandwidth for both the client and the repository provider. The `metadata_expire` (see [dnf.conf\(5\)](#)) repository configuration option is used by DNF to determine whether a particular local copy of repository data is due to be re-synced. It is crucial that the repository providers set the option well, namely to a value where it is guaranteed that if particular metadata was available in time **T** on the server, then all packages it references will still be available for download from the server in time **T** + `metadata_expire`.

To further reduce the bandwidth load, some of the commands where having up-to-date metadata is not critical (e.g. the `list` command) do not look at whether a repository is expired and whenever any version of it is locally available to the user's account, it will be used. For non-root use, see also the `--cacheonly` switch. Note that in all situations the user can force synchronization of all enabled repositories with the `--refresh` switch.

CONFIGURATION FILES REPLACEMENT POLICY

[top](#)

The updated packages could replace the old modified configuration files with the new ones or keep the older files. Neither of the files are actually replaced. To the conflicting ones RPM gives additional suffix to the origin name. Which file should maintain the true name after transaction is not controlled by package manager but is specified by each package itself, following packaging guideline.

FILES

[top](#)

Cache Files
`/var/cache/dnf`

Main Configuration
`/etc/dnf/dnf.conf`

Repository
`/etc/yum.repos.d/`

SEE ALSO

[top](#)

- [dnf.conf\(5\)](#), *DNF Configuration Reference*
- [dnf-PLUGIN\(8\)](#) for documentation on DNF plugins.
- [dnf.modularity\(7\)](#), *Modularity overview*.
- [dnf-transaction-json\(5\)](#), *Stored Transaction JSON Format Specification*.
- DNF project homepage (-
<https://github.com/rpm-software-management/dnf/>)
- How to report a bug (-
<https://github.com/rpm-software-management/dnf/wiki/Bug-Reporting>)
- YUM project homepage (<http://yum.baseurl.org/>)

AUTHOR [top](#)

See AUTHORS in DNF source distribution.

COPYRIGHT [top](#)

2012-2020, Red Hat, Licensed under GPLv2+

COLOPHON [top](#)

This page is part of the [dnf](#) (DNF Package Manager) project. Information about the project can be found at (<https://github.com/rpm-software-management/dnf>). It is not known how to report bugs for this man page; if you know, please send a mail to man-pages@man7.org. This page was obtained from the project's upstream Git repository (<https://github.com/rpm-software-management/dnf.git>) on 2022-12-17. (At that time, the date of the most recent commit that was found in the repository was 2022-11-30.) If you discover any rendering problems in this HTML version of the page, or you believe there is a better or more up-to-date source for the page, or you have corrections or improvements to the information in this COLOPHON (which is *not* part of the original manual page), send a mail to man-pages@man7.org

4.14.0

Dec 17, 2022

DNF(8)

Pages that refer to this page: [systemd-nspawn\(1\)](#), [dnf.conf\(5\)](#), [yum.conf\(5\)](#)

HTML rendering created 2022-12-18 by Michael Kerrisk, author of *The Linux Programming Interface*, maintainer of the [Linux man-pages](#) project.

For details of in-depth Linux/UNIX system programming training courses that I teach, look [here](#).

Hosting by [jambit GmbH](#).



NAME

do-release-upgrade – upgrade operating system to latest release

SYNOPSIS

do-release-upgrade [*options*]

DESCRIPTION

Upgrade the operating system to the latest release from the command-line. This is the preferred command if the machine has no graphic environment or if the machine is to be upgraded over a remote connection.

OPTIONS

-h, --help

show help message and exit

-d, --devel-release

If using the latest supported release, upgrade to the development release

-p, --proposed

Try upgrading to the latest release using the upgrader from Ubuntu-proposed

-m MODE, --mode=MODE

Run in a special upgrade mode. Currently "desktop" for regular upgrades of a desktop system and "server" for server systems are supported.

-f FRONTEND, --frontend=FRONTEND

Run the specified frontend

SEE ALSO

update-manager(8), apt-get(8)

NAME

fsck.fat – check and repair MS-DOS FAT filesystems

SYNOPSIS

fsck.fat [*OPTIONS*] *DEVICE*

DESCRIPTION

fsck.fat verifies the consistency of MS-DOS filesystems and optionally tries to repair them.

The following filesystem problems can be corrected (in this order):

- FAT contains invalid cluster numbers. Cluster is changed to EOF.
- File's cluster chain contains a loop. The loop is broken.
- Bad clusters (read errors). The clusters are marked bad and they are removed from files owning them. This check is optional.
- Directories with a large number of bad entries (probably corrupt). The directory can be deleted.
- Files . and .. are non-directories. They can be deleted or renamed.
- Directories . and .. in root directory. They are deleted.
- Bad filenames. They can be renamed.
- Duplicate directory entries. They can be deleted or renamed.
- Directories with non-zero size field. Size is set to zero.
- Directory . does not point to parent directory. The start pointer is adjusted.
- Directory .. does not point to parent of parent directory. The start pointer is adjusted.
- . and .. are not the two first entries in a non-root directory. The entries are created, moving occupied slots if necessary.
- Start cluster number of a file is invalid. The file is truncated.
- File contains bad or free clusters. The file is truncated.
- File's cluster chain is longer than indicated by the size fields. The file is truncated.
- Two or more files share the same cluster(s). All but one of the files are truncated. If the file being truncated is a directory file that has already been read, the filesystem check is restarted after truncation.
- File's cluster chain is shorter than indicated by the size fields. The file is truncated.
- Volume label in root directory or label in boot sector is invalid. Invalid labels are removed.
- Volume label in root directory and label in boot sector are different. Volume label from root directory is copied to boot sector.
- Clusters are marked as used but are not owned by a file. They are marked as free.

Additionally, the following problems are detected, but not repaired:

- Invalid parameters in boot sector

When **fsck.fat** checks a filesystem, it accumulates all changes in memory and performs them only after all checks are complete. This can be disabled with the **-w** option.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

There are some minor differences in Atari format: Some boot sector fields are interpreted slightly different, and the special FAT entries for end-of-file and bad cluster can be different. Under MS-DOS 0xffff8 is used for EOF and Atari employs 0xffff by default, but both systems recognize all values from 0xffff8–0xffff as end-of-file. MS-DOS uses only 0xffff7 for bad clusters, where on Atari values 0xffff0–0xffff7 are for this

purpose (but the standard value is still 0xffff7).

OPTIONS

-a Automatically repair the filesystem. No user intervention is necessary. Whenever there is more than one method to solve a problem, the least destructive approach is used.

-A Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.

-b Make read-only boot sector check.

-c PAGE

Use DOS codepage *PAGE* to decode short file names. By default codepage 850 is used.

-d PATH

Delete the specified file. If more than one file with that name exist, the first one is deleted. This option can be given more than once.

-f Salvage unused cluster chains to files. By default, unused clusters are added to the free disk space except in auto mode (**-a**).

-F NUM

Specify FAT table *NUM* for filesystem access. By default value 0 is assumed and then the first uncorrupted FAT table is chosen. Uncorrupted means that FAT table has valid first cluster. If default value 0 is used and all FAT tables are corrupted then **fsck.fat** gives up and does not try to repair FAT filesystem. If non-zero *NUM* value is specified then **fsck.fat** uses FAT table *NUM* for repairing FAT filesystem. If FAT table *NUM* has corrupted first cluster then **fsck.fat** will repair it. In any case, if FAT filesystem has more FAT tables then repaired content of chosen FAT table is copied to other FAT tables. To repair corrupted first cluster it is required to call **fsck.fat** with non-zero *NUM* value.

-l List path names of files being processed.

-n No-operation mode: non-interactively check for errors, but don't write anything to the filesystem.

-p Same as **-a**, for compatibility with other *fsck.

-r Interactively repair the filesystem. The user is asked for advice whenever there is more than one approach to fix an inconsistency. This is the default mode and the option is only retained for backwards compatibility.

-S Consider short (8.3) file names with spaces in the middle to be invalid, like previous versions of this program did. While such file names are not forbidden by the FAT specification, and were never treated as errors by Microsoft file system checking tools, many DOS programs are unable to handle files with such names. Using this option can make them accessible to these programs.

Short file names which *start* with a space are considered invalid regardless of this option's setting.

Previous versions of this program exceptionally treated *EA DATA. SF* and *WP ROOT. SF* as valid short names; using this option does not preserve that exception.

-t Mark unreadable clusters as bad.

-u PATH

Try to undelete the specified file. **fsck.fat** tries to allocate a chain of contiguous unallocated clusters beginning with the start cluster of the undeleted file. This option can be given more than once.

-U Consider lowercase volume and boot label as invalid and allow only uppercase characters. Such labels are forbidden by the FAT specification, but they are widely used by Linux tools. Moreover MS-DOS and Windows systems do not have problems to read them. Therefore volume and boot labels with lowercase characters are by default permitted.

-v Verbose mode. Generates slightly more output.

-V Perform a verification pass. The filesystem check is repeated after the first run. The second pass should never report any fixable errors. It may take considerably longer than the first pass, because the first pass may have generated long list of modifications that have to be scanned for each disk read.

--variant *TYPE*

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

-w Write changes to disk immediately.

-y Same as **-a** (automatically repair filesystem) for compatibility with other fsck tools.

--help

Display help message describing usage and options then exit.

EXIT STATUS

0 No recoverable errors have been detected.

1 Recoverable errors have been detected or **fsck.fat** has discovered an internal inconsistency.

2 Usage error. **fsck.fat** did not access the filesystem.

FILES

fsck0000.rec, *fsck0001.rec*, ...

When recovering from a corrupted filesystem, **fsck.fat** dumps recovered data into files named *fsckNNNN.rec* in the top level directory of the filesystem.

BUGS

- Does not remove entirely empty directories.
- Should give more diagnostic messages.
- Undeleting files should use a more sophisticated algorithm.

SEE ALSO

fatlabel(8), **mkfs.fat(8)**

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

dpkg – package manager for Debian

SYNOPSIS

dpkg [*option...*] *action*

WARNING

This manual is intended for users wishing to understand **dpkg**'s command line options and package states in more detail than that provided by **dpkg --help**.

It should *not* be used by package maintainers wishing to understand how **dpkg** will install their packages. The descriptions of what **dpkg** does when installing and removing packages are particularly inadequate.

DESCRIPTION

dpkg is a medium-level tool to install, build, remove and manage Debian packages. The primary and more user-friendly front-end for **dpkg** as a CLI (command-line interface) is **apt(8)** and as a TUI (terminal user interface) is **aptitude(8)**. **dpkg** itself is controlled entirely via command line parameters, which consist of exactly one action and zero or more options. The action-parameter tells **dpkg** what to do and options control the behavior of the action in some way.

dpkg can also be used as a front-end to **dpkg-deb(1)** and **dpkg-query(1)**. The list of supported actions can be found later on in the ACTIONS section. If any such action is encountered **dpkg** just runs **dpkg-deb** or **dpkg-query** with the parameters given to it, but no specific options are currently passed to them, to use any such option the back-ends need to be called directly.

INFORMATION ABOUT PACKAGES

dpkg maintains some usable information about available packages. The information is divided in three classes: **states**, **selection states** and **flags**. These values are intended to be changed mainly with **dselect**.

Package states

not-installed

The package is not installed on your system.

config-files

Only the configuration files or the **postrm** script and the data it needs to remove of the package exist on the system.

half-installed

The installation of the package has been started, but not completed for some reason.

unpacked

The package is unpacked, but not configured.

half-configured

The package is unpacked and configuration has been started, but not yet completed for some reason.

triggers-awaited

The package awaits trigger processing by another package.

triggers-pending

The package has been triggered.

installed

The package is correctly unpacked and configured.

Package selection states

install

The package is selected for installation.

hold

A package marked to be on **hold** is kept on the same version, that is, no automatic new installs, upgrades or removals will be performed on them, unless these actions are requested explicitly, or are permitted to be done automatically with the **--force-hold** option.

deinstall

The package is selected for deinstallation (i.e. we want to remove all files, except configuration files).

purge

The package is selected to be purged (i.e. we want to remove everything from system directories, even configuration files).

unknown

The package selection is unknown. A package that is also in **not-installed** state, and with an **ok** flag will be forgotten in the next database store.

Package flags

ok A package marked **ok** is in a known state, but might need further processing.

reinstreq

A package marked **reinstreq** is broken and requires reinstallation. These packages cannot be removed, unless forced with option **--force-remove-reinstreq**.

ACTIONS**-i, --install package-file...**

Install the package. If **--recursive** or **-R** option is specified, *package-file* must refer to a directory instead.

Installation consists of the following steps:

1. Extract the control files of the new package.
2. If another version of the same package was installed before the new installation, execute *prerm* script of the old package.
3. Run *preinst* script, if provided by the package.
4. Unpack the new files, and at the same time back up the old files, so that if something goes wrong, they can be restored.
5. If another version of the same package was installed before the new installation, execute the *postrm* script of the old package. Note that this script is executed after the *preinst* script of the new package, because new files are written at the same time old files are removed.
6. Configure the package. See **--configure** for detailed information about how this is done.

--unpack package-file...

Unpack the package, but don't configure it. If **--recursive** or **-R** option is specified, *package-file* must refer to a directory instead.

--configure package...|-a|--pending

Configure a package which has been unpacked but not yet configured. If **-a** or **--pending** is given instead of *package*, all unpacked but unconfigured packages are configured.

To reconfigure a package which has already been configured, try the **dpkg-reconfigure(8)** command instead.

Configuring consists of the following steps:

1. Unpack the conffiles, and at the same time back up the old conffiles, so that they can be restored if something goes wrong.
2. Run *postinst* script, if provided by the package.

--triggers-only package...|-a|--pending

Processes only triggers (since dpkg 1.14.17). All pending triggers will be processed. If package names are supplied only those packages' triggers will be processed, exactly once each where necessary. Use of this option may leave packages in the improper **triggers-awaited** and **triggers-pending** states. This can be fixed later by running: **dpkg --configure --pending**.

-r, --remove package...|-a|--pending

Remove an installed package. This removes everything except conffiles and other data cleaned up by the *postrm* script, which may avoid having to reconfigure the package if it is reinstalled later (conffiles are configuration files that are listed in the *DEBIAN/conffiles* control file). If there is no *DEBIAN/conffiles* control file nor *DEBIAN/postrm* script, this command is equivalent to calling **--purge**. If **-a** or **--pending** is given instead of a package name, then all packages unpacked, but marked to be removed in file */var/lib/dpkg/status*, are removed.

Removing of a package consists of the following steps:

- 1.** Run *prerm* script
- 2.** Remove the installed files
- 3.** Run *postrm* script

-P, --purge package...|-a|--pending

Purge an installed or already removed package. This removes everything, including conffiles, and anything else cleaned up from *postrm*. If **-a** or **--pending** is given instead of a package name, then all packages unpacked or removed, but marked to be purged in file */var/lib/dpkg/status*, are purged.

Note: Some configuration files might be unknown to **dpkg** because they are created and handled separately through the configuration scripts. In that case, **dpkg** won't remove them by itself, but the package's *postrm* script (which is called by **dpkg**), has to take care of their removal during purge. Of course, this only applies to files in system directories, not configuration files written to individual users' home directories.

Purging of a package consists of the following steps:

- 1.** Remove the package, if not already removed. See **--remove** for detailed information about how this is done.
- 2.** Run *postrm* script.

-V, --verify [package-name...]

Verifies the integrity of *package-name* or all packages if omitted, by comparing information from the files installed by a package with the files metadata information stored in the **dpkg** database (since *dpkg* 1.17.2). The origin of the files metadata information in the database is the binary packages themselves. That metadata gets collected at package unpack time during the installation process.

Currently the only functional check performed is an md5sum verification of the file contents against the stored value in the files database. It will only get checked if the database contains the file md5sum. To check for any missing metadata in the database, the **--audit** command can be used.

The output format is selectable with the **--verify-format** option, which by default uses the **rpm** format, but that might change in the future, and as such, programs parsing this command output should be explicit about the format they expect.

-C, --audit [package-name...]

Performs database sanity and consistency checks for *package-name* or all packages if omitted (per package checks since *dpkg* 1.17.10). For example, searches for packages that have been installed only partially on your system or that have missing, wrong or obsolete control data or files. **dpkg** will suggest what to do with them to get them fixed.

--update-avail [Packages-file]**--merge-avail [Packages-file]**

Update **dpkg**'s and **dselect**'s idea of which packages are available. With action **--merge-avail**, old information is combined with information from *Packages-file*. With action **--update-avail**, old information is replaced with the information in the *Packages-file*. The *Packages-file* distributed with Debian is simply named «*Packages*». If the *Packages-file* argument is missing or named «--» then it will be read from standard input (since *dpkg* 1.17.7). **dpkg** keeps its record of available packages in */var/lib/dpkg/available*.

A simpler one-shot command to retrieve and update the *available* file is **dselect update**. Note that this file is mostly useless if you don't use **dselect** but an APT-based frontend: APT has its own system to keep track of available packages.

-A, --record-avail *package-file...*

Update **dpkg** and **dselect**'s idea of which packages are available with information from the package *package-file*. If **--recursive** or **-R** option is specified, *package-file* must refer to a directory instead.

--forget-old-unavail

Now **obsolete** and a no-op as **dpkg** will automatically forget uninstalled unavailable packages (since **dpkg** 1.15.4), but only those that do not contain user information such as package selections.

--clear-avail

Erase the existing information about what packages are available.

--get-selections [*package-name-pattern...*]

Get list of package selections, and write it to stdout. Without a pattern, non-installed packages (i.e. those which have been previously purged) will not be shown.

--set-selections

Set package selections using file read from stdin. This file should be in the format "*package state*", where state is one of **install**, **hold**, **deinstall** or **purge**. Blank lines and comment lines beginning with '#' are also permitted.

The *available* file needs to be up-to-date for this command to be useful, otherwise unknown packages will be ignored with a warning. See the **--update-avail** and **--merge-avail** commands for more information.

--clear-selections

Set the requested state of every non-essential package to deinstall (since **dpkg** 1.13.18). This is intended to be used immediately before **--set-selections**, to deinstall any packages not in list given to **--set-selections**.

--yet-to-unpack

Searches for packages selected for installation, but which for some reason still haven't been installed.

Note: This command makes use of both the available file and the package selections.

--predep-package

Print a single package which is the target of one or more relevant pre-dependencies and has itself no unsatisfied pre-dependencies.

If such a package is present, output it as a Packages file entry, which can be massaged as appropriate.

Note: This command makes use of both the available file and the package selections.

Returns 0 when a package is printed, 1 when no suitable package is available and 2 on error.

--add-architecture *architecture*

Add *architecture* to the list of architectures for which packages can be installed without using **--force-architecture** (since **dpkg** 1.16.2). The architecture **dpkg** is built for (i.e. the output of **--print-architecture**) is always part of that list.

--remove-architecture *architecture*

Remove *architecture* from the list of architectures for which packages can be installed without using **--force-architecture** (since **dpkg** 1.16.2). If the architecture is currently in use in the database then the operation will be refused, except if **--force-architecture** is specified. The architecture **dpkg** is built for (i.e. the output of **--print-architecture**) can never be removed from that list.

--print-architecture

Print architecture of packages **dpkg** installs (for example, "i386").

--print-foreign-architectures

Print a newline-separated list of the extra architectures **dpkg** is configured to allow packages to be installed for (since dpkg 1.16.2).

--assert-help

Give help about the **--assert-*feature*** options (since dpkg 1.21.0).

--assert-*feature*

Asserts that **dpkg** supports the requested feature. Returns 0 if the feature is fully supported, 1 if the feature is known but **dpkg** cannot provide support for it yet, and 2 if the feature is unknown. The current list of assertable features is:

support-predepends

Supports the **Pre-Depends** field (since dpkg 1.1.0).

working-epoch

Supports epochs in version strings (since dpkg 1.4.0.7).

long-filenames

Supports long filenames in **deb(5)** archives (since dpkg 1.4.1.17).

multi-conrep

Supports multiple **Conflicts** and **Replaces** (since dpkg 1.4.1.19).

multi-arch

Supports multi-arch fields and semantics (since dpkg 1.16.2).

versioned-provides

Supports versioned **Provides** (since dpkg 1.17.11).

protected-field

Supports the **Protected** field (since dpkg 1.20.1).

--validate-*thing string*

Validate that the *thing string* has a correct syntax (since dpkg 1.18.16). Returns 0 if the *string* is valid, 1 if the *string* is invalid but might be accepted in lax contexts, and 2 if the *string* is invalid. The current list of validatable *things* is:

pkgname

Validates the given package name (since dpkg 1.18.16).

trigname

Validates the given trigger name (since dpkg 1.18.16).

archname

Validates the given architecture name (since dpkg 1.18.16).

version

Validates the given version (since dpkg 1.18.16).

--compare-versions *ver1 op ver2*

Compare version numbers, where *op* is a binary operator. **dpkg** returns true (**0**) if the specified condition is satisfied, and false (**1**) otherwise. There are two groups of operators, which differ in how they treat an empty *ver1* or *ver2*. These treat an empty version as earlier than any version: **lt** **le** **eq** **ne** **ge** **gt**. These treat an empty version as later than any version: **lt-nl** **le-nl** **ge-nl** **gt-nl**. These are provided only for compatibility with control file syntax: <<< <= = >= >>>. The < and > operators are obsolete and should **not** be used, due to confusing semantics. To illustrate: **0.1 < 0.1** evaluates to true.

-?, --help

Display a brief help message.

--force-help

Give help about the **--force-*thing*** options.

-Dh, --debug=help

Give help about debugging options.

--version

Display **dpkg** version information.

When used with **--robot**, the output will be the program version number in a dotted numerical format, with no newline.

dpkg-deb actions

See **dpkg-deb(1)** for more information about the following actions, and other actions and options not exposed by the **dpkg** front-end.

-b, --build directory [archive|directory]

Build a deb package.

-c, --contents archive

List contents of a deb package.

-e, --control archive [directory]

Extract control-information from a package.

-x, --extract archive directory

Extract the files contained by package.

-X, --vextract archive directory

Extract and display the filenames contained by a package.

-f, --field archive [control-field...]

Display control field(s) of a package.

--ctrl-tarfile archive

Output the control tar-file contained in a Debian package.

--fsys-tarfile archive

Output the filesystem tar-file contained by a Debian package.

-I, --info archive [control-file...]

Show information about a package.

dpkg-query actions

See **dpkg-query(1)** for more information about the following actions, and other actions and options not exposed by the **dpkg** front-end.

-l, --list package-name-pattern...

List packages matching given pattern.

-s, --status package-name...

Report status of specified package.

-L, --listfiles package-name...

List files installed to your system from *package-name*.

-S, --search filename-search-pattern...

Search for a filename from installed packages.

-p, --print-avail package-name...

Display details about *package-name*, as found in */var/lib/dpkg/available*. Users of APT-based frontends should use **apt show package-name** instead.

OPTIONS

All options can be specified both on the command line and in the **dpkg** configuration file */etc/dpkg/dpkg.cfg* or fragment files (with names matching this shell pattern '[0-9a-zA-Z_-]*') on the configuration directory */etc/dpkg/dpkg.cfg.d/*. Each line in the configuration file is either an option (exactly the same as the command line option but without leading hyphens) or a comment (if it starts with a '#').

--abort-after=number

Change after how many errors **dpkg** will abort. The default is 50.

-B, --auto-deconfigure

When a package is removed, there is a possibility that another installed package depended on the removed package. Specifying this option will cause automatic deconfiguration of the package which depended on the removed package.

-D{octal}, --debug{octal}

Switch debugging on. *octal* is formed by bitwise-ORing desired values together from the list below (note that these values may change in future releases). **-Dh** or **--debug=help** display these debugging values.

Number	Description
1	Generally helpful progress information
2	Invocation and status of maintainer scripts
10	Output for each file processed
100	Lots of output for each file processed
20	Output for each configuration file
200	Lots of output for each configuration file
40	Dependencies and conflicts
400	Lots of dependencies/conflicts output
10000	Trigger activation and processing
20000	Lots of output regarding triggers
40000	Silly amounts of output regarding triggers
1000	Lots of drivel about e.g. the dpkg/info dir
2000	Insane amounts of drivel

--force-things**--no-force-things, --refuse-things**

Force or refuse (**no-force** and **refuse** mean the same thing) to do some things. *things* is a comma separated list of things specified below. **--force-help** displays a message describing them. Things marked with (*) are forced by default.

Warning: These options are mostly intended to be used by experts only. Using them without fully understanding their effects may break your whole system.

all: Turns on (or off) all force options.

downgrade(*): Install a package, even if newer version of it is already installed.

Warning: At present **dpkg** does not do any dependency checking on downgrades and therefore will not warn you if the downgrade breaks the dependency of some other package. This can have serious side effects, downgrading essential system components can even make your whole system unusable. Use with care.

configure-any: Configure also any unpacked but unconfigured packages on which the current package depends.

hold: Allow automatic installs, upgrades or removals of packages even when marked to be on “hold”.
Note: This does not prevent these actions when requested explicitly.

remove-reinstreq: Remove a package, even if it’s broken and marked to require reinstallation. This may, for example, cause parts of the package to remain on the system, which will then be forgotten by **dpkg**.

remove-protected: Remove, even if the package is considered protected (since dpkg 1.20.1). Protected packages contain mostly important system boot infrastructure. Removing them might cause the whole system to be unable to boot, so use with caution.

remove-essential: Remove, even if the package is considered essential. Essential packages contain

mostly very basic Unix commands. Removing them might cause the whole system to stop working, so use with caution.

depends: Turn all dependency problems into warnings. This affects the **Pre-Depends** and **Depends** fields.

depends-version: Don't care about versions when checking dependencies. This affects the **Pre-Depends** and **Depends** fields.

breaks: Install, even if this would break another package (since dpkg 1.14.6). This affects the **Breaks** field.

conflicts: Install, even if it conflicts with another package. This is dangerous, for it will usually cause overwriting of some files. This affects the **Conflicts** field.

confmiss: Always install the missing conffile without prompting. This is dangerous, since it means not preserving a change (removing) made to the file.

confnew: If a conffile has been modified and the version in the package did change, always install the new version without prompting, unless the **--force-confdef** is also specified, in which case the default action is preferred.

confold: If a conffile has been modified and the version in the package did change, always keep the old version without prompting, unless the **--force-confdef** is also specified, in which case the default action is preferred.

confdef: If a conffile has been modified and the version in the package did change, always choose the default action without prompting. If there is no default action it will stop to ask the user unless **--force-confnew** or **--force-confold** is also given, in which case it will use that to decide the final action.

confask: If a conffile has been modified always offer to replace it with the version in the package, even if the version in the package did not change (since dpkg 1.15.8). If any of **--force-confnew**, **--force-confold**, or **--force-confdef** is also given, it will be used to decide the final action.

overwrite: Overwrite one package's file with another's file.

overwrite-dir: Overwrite one package's directory with another's file.

overwrite-diverted: Overwrite a diverted file with an undiverted version.

statoverride-add: Overwrite an existing stat override when adding it (since dpkg 1.19.5).

statoverride-remove: Ignore a missing stat override when removing it (since dpkg 1.19.5).

security-mac^(*): Use platform-specific Mandatory Access Controls (MAC) based security when installing files into the filesystem (since dpkg 1.19.5). On Linux systems the implementation uses SELinux.

unsafe-io: Do not perform safe I/O operations when unpacking (since dpkg 1.15.8.6). Currently this implies not performing file system syncs before file renames, which is known to cause substantial performance degradation on some file systems, unfortunately the ones that require the safe I/O on the first place due to their unreliable behaviour causing zero-length files on abrupt system crashes.

Note: For ext4, the main offender, consider using instead the mount option **nodelalloc**, which will fix both the performance degradation and the data safety issues, the latter by making the file system not produce zero-length files on abrupt system crashes with any software not doing syncs before atomic renames.

Warning: Using this option might improve performance at the cost of losing data, use with care.

script-chrootless: Run maintainer scripts without **chroot**(2)ing into **instdir** even if the package does not support this mode of operation (since dpkg 1.18.5).

Warning: This can destroy your host system, use with extreme care.

architecture: Process even packages with wrong or no architecture.

bad-version: Process even packages with wrong versions (since dpkg 1.16.1).

bad-path: PATH is missing important programs, so problems are likely.

not-root: Try to (de)install things even when not root.

bad-verify: Install a package even if it fails authenticity check.

--ignore-depends=package,...

Ignore dependency-checking for specified packages (actually, checking is performed, but only warnings about conflicts are given, nothing else). This affects the **Pre-Depends**, **Depends** and **Breaks** fields.

--no-act, --dry-run, --simulate

Do everything which is supposed to be done, but don't write any changes. This is used to see what would happen with the specified action, without actually modifying anything.

Be sure to give **--no-act** before the action-parameter, or you might end up with undesirable results. (e.g. **dpkg --purge foo --no-act** will first purge package "foo" and then try to purge package "--no-act", even though you probably expected it to actually do nothing).

-R, --recursive

Recursively handle all regular files matching pattern ***.deb** found at specified directories and all of its subdirectories. This can be used with **-i**, **-A**, **--install**, **--unpack** and **--record-avail** actions.

-G Don't install a package if a newer version of the same package is already installed. This is an alias of **--refuse-downgrade**.

--admindir=dir

Set the administrative directory to *directory*. This directory contains many files that give information about status of installed or uninstalled packages, etc. Defaults to **<</var/lib/dpkg>>**.

--instdir=dir

Set the installation directory, which refers to the directory where packages are to be installed. **instdir** is also the directory passed to **chroot(2)** before running package's installation scripts, which means that the scripts see **instdir** as a root directory. Defaults to **<</>>**.

--root=dir

Set the root directory to **directory**, which sets the installation directory to **<<dir>>** and the administrative directory to **<<dir/var/lib/dpkg>>**.

-O, --selected-only

Only process the packages that are selected for installation. The actual marking is done with **dselect** or by **dpkg**, when it handles packages. For example, when a package is removed, it will be marked selected for deinstallation.

-E, --skip-same-version

Don't install the package if the same version of the package is already installed.

--pre-invoke=command

--post-invoke=command

Set an invoke hook *command* to be run via "sh -c" before or after the **dpkg** run for the *unpack*, *configure*, *install*, *triggers-only*, *remove*, *purge*, *add-architecture* and *remove-architecture* **dpkg** actions (since dpkg 1.15.4; *add-architecture* and *remove-architecture* actions since dpkg 1.17.19). This option can be specified multiple times. The order the options are specified is preserved, with the ones from the configuration files taking precedence. The environment variable **DPKG_HOOK_ACTION** is set for the hooks to the current **dpkg** action.

Note: Front-ends might call **dpkg** several times per invocation, which might run the hooks more times than expected.

--path-exclude=glob-pattern
--path-include=glob-pattern

Set *glob-pattern* as a path filter, either by excluding or re-including previously excluded paths matching the specified patterns during install (since dpkg 1.15.8).

Warning: Take into account that depending on the excluded paths you might completely break your system, use with caution.

The glob patterns use the same wildcards used in the shell, were '*' matches any sequence of characters, including the empty string and also '/'. For example, «*/usr/*/README*» matches «*/usr/share/doc/package/README*». As usual, '?' matches any single character (again, including '/'). And '[' starts a character class, which can contain a list of characters, ranges and complementations. See **glob(7)** for detailed information about globbing. **Note:** The current implementation might re-include more directories and symlinks than needed, in particular when there is a more specific re-inclusion, to be on the safe side and avoid possible unpack failures; future work might fix this.

This can be used to remove all paths except some particular ones; a typical case is:

```
--path-exclude=/usr/share/doc/*
--path-include=/usr/share/doc/*/copyright
```

to remove all documentation files except the copyright files.

These two options can be specified multiple times, and interleaved with each other. Both are processed in the given order, with the last rule that matches a file name making the decision.

The filters are applied when unpacking the binary packages, and as such only have knowledge of the type of object currently being filtered (e.g. a normal file or a directory) and have not visibility of what objects will come next. Because these filters have side effects (in contrast to **find(1)** filters), excluding an exact pathname that happens to be a directory object like */usr/share/doc* will not have the desired result, and only that pathname will be excluded (which could be automatically reincluded if the code sees the need). Any subsequent files contained within that directory will fail to unpack.

Hint: make sure the globs are not expanded by your shell.

--verify-format format-name

Sets the output format for the **--verify** command (since dpkg 1.17.2).

The only currently supported output format is **rpm**, which consists of a line for every path that failed any check. These lines have the following format:

```
missing [c] pathname [(error-message)]
??5????? [c] pathname
```

The first 9 characters are used to report the checks result, either a literal **missing** when the file is not present or its metadata cannot be fetched, or one of the following special characters that report the result for each check:

'?' Implies the check could not be done (lack of support, file permissions, etc).

'.' Implies the check passed.

'A-Za-z0-9'

Implies a specific check failed. The following positions and alphanumeric characters are currently supported:

1 '?'

These checks are currently not supported, will always be '?'.

2 'M'

The file mode check failed (since dpkg 1.21.0). Because pathname metadata is currently not tracked, this check can only be partially emulated via a very simple heuristic for pathnames that have a known digest, which implies they should be regular files, where the check will

fail if the pathname is not a regular file on the filesystem. This check will currently never succeed as it does not have enough information available.

3 ‘5’

The digest check failed, which means the file contents have changed.

4–9 ‘?’

These checks are currently not supported, will always be ‘?’.

The line is followed by a space and an attribute character. The following attribute character is supported:

‘c’ The pathname is a conffile.

Finally followed by another space and the pathname.

In case the entry was of the **missing** type, and the file was not actually present on the filesystem, then the line is followed by a space and the error message enclosed within parenthesis.

--status-fd *n*

Send machine-readable package status and progress information to file descriptor *n*. This option can be specified multiple times. The information is generally one record per line, in one of the following forms:

status: *package*: *status*

Package status changed; *status* is as in the status file.

status: *package* : **error** : *extended-error-message*

An error occurred. Any possible newlines in *extended-error-message* will be converted to spaces before output.

status: *file* : **conffile-prompt** : ‘*real-old*’ ‘*real-new*’ *useredited distedited*

User is being asked a conffile question.

processing: *stage*: *package*

Sent just before a processing stage starts. *stage* is one of **upgrade**, **install** (both sent before unpacking), **configure**, **trigproc**, **disappear**, **remove**, **purge**.

--status-logger=*command*

Send machine-readable package status and progress information to the shell *command*’s standard input, to be run via “sh –c” (since dpkg 1.16.0). This option can be specified multiple times. The output format used is the same as in --status-fd.

--log=*filename*

Log status change updates and actions to *filename*, instead of the default */var/log/dpkg.log*. If this option is given multiple times, the last filename is used. Log messages are of the form:

YYYY-MM-DD HH:MM:SS **startup** *type command*

For each dpkg invocation where *type* is **archives** (with a *command* of **unpack** or **install**) or **packages** (with a *command* of **configure**, **triggers-only**, **remove** or **purge**).

YYYY-MM-DD HH:MM:SS **status** *state pkg installed-version*

For status change updates.

YYYY-MM-DD HH:MM:SS *action pkg installed-version available-version*

For actions where *action* is one of **install**, **upgrade**, **configure**, **trigproc**, **disappear**, **remove** or **purge**.

YYYY-MM-DD HH:MM:SS **conffile** *filename decision*

For conffile changes where *decision* is either **install** or **keep**.

--robot

Use a machine-readable output format. This provides an interface for programs that need to parse the output of some of the commands that do not otherwise emit a machine-readable output format. No localization will be used, and the output will be modified to make it easier to parse.

The only currently supported command is **--version**.

--no-pager

Disables the use of any pager when showing information (since dpkg 1.19.2).

--no-debsig

Do not try to verify package signatures.

--no-triggers

Do not run any triggers in this run (since dpkg 1.14.17), but activations will still be recorded. If used with **--configure package** or **--triggers-only package** then the named package postinst will still be run even if only a triggers run is needed. Use of this option may leave packages in the improper **triggers-awaited** and **triggers-pending** states. This can be fixed later by running: **dpkg --configure --pending**.

--triggers

Cancels a previous **--no-triggers** (since dpkg 1.14.17).

EXIT STATUS

- 0** The requested action was successfully performed. Or a check or assertion command returned true.
- 1** A check or assertion command returned false.
- 2** Fatal or unrecoverable error due to invalid command-line usage, or interactions with the system, such as accesses to the database, memory allocations, etc.

ENVIRONMENT**External environment****PATH**

This variable is expected to be defined in the environment and point to the system paths where several required programs are to be found. If it's not set or the programs are not found, **dpkg** will abort.

HOME

If set, **dpkg** will use it as the directory from which to read the user specific configuration file.

TMPDIR

If set, **dpkg** will use it as the directory in which to create temporary files and directories.

SHELL

The program **dpkg** will execute when starting a new interactive shell, or when spawning a command via a shell.

PAGER**DPKG_PAGER**

The program **dpkg** will execute when running a pager, which will be executed with «**\$SHELL -c**», for example when displaying the conffile differences. If **SHELL** is not set, «**sh**» will be used instead. The **DPKG_PAGER** overrides the **PAGER** environment variable (since dpkg 1.19.2).

DPKG_COLORS

Sets the color mode (since dpkg 1.18.5). The currently accepted values are: **auto** (default), **always** and **never**.

DPKG_FORCE

Sets the force flags (since dpkg 1.19.5). When this variable is present, no built-in force defaults will be applied. If the variable is present but empty, all force flags will be disabled.

DPKG_ADMINDIR

If set and the **--admindir** or **--root** options have not been specified, it will be used as the **dpkg** administrative directory (since dpkg 1.20.0).

DPKG_FRONTEND_LOCKED

Set by a package manager frontend to notify dpkg that it should not acquire the frontend lock (since dpkg 1.19.1).

Internal environment**LESS**

Defined by **dpkg** to “**-FRSXMQ**”, if not already set, when spawning a pager (since dpkg 1.19.2). To change the default behavior, this variable can be preset to some other value including an empty string, or the **PAGER** or **DPKG_PAGER** variables can be set to disable specific options with «**-+**», for example **DPKG_PAGER=“less -+F”**.

DPKG_ROOT

Defined by **dpkg** on the maintainer script environment to indicate which installation to act on (since dpkg 1.18.5). The value is intended to be prepended to any path maintainer scripts operate on. During normal operation, this variable is empty. When installing packages into a different **instdir**, **dpkg** normally invokes maintainer scripts using **chroot(2)** and leaves this variable empty, but if **--force-script-chrootless** is specified then the **chroot(2)** call is skipped and **instdir** is non-empty.

DPKG_ADMINDIR

Defined by **dpkg** on the maintainer script environment to indicate the **dpkg** administrative directory to use (since dpkg 1.16.0). This variable is always set to the current **--admindir** value.

DPKG_FORCE

Defined by **dpkg** on the subprocesses environment to all the currently enabled force option names separated by commas (since dpkg 1.19.5).

DPKG_SHELL_REASON

Defined by **dpkg** on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Current valid value: **conffile-prompt**.

DPKG_CONFFILE_OLD

Defined by **dpkg** on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Contains the path to the old conffile.

DPKG_CONFFILE_NEW

Defined by **dpkg** on the shell spawned on the conffile prompt to examine the situation (since dpkg 1.15.6). Contains the path to the new conffile.

DPKG_HOOK_ACTION

Defined by **dpkg** on the shell spawned when executing a hook action (since dpkg 1.15.4). Contains the current **dpkg** action.

DPKG_RUNNING_VERSION

Defined by **dpkg** on the maintainer script environment to the version of the currently running **dpkg** instance (since dpkg 1.14.17).

DPKG_MAINTSCRIPT_PACKAGE

Defined by **dpkg** on the maintainer script environment to the (non-arch-qualified) package name being handled (since dpkg 1.14.17).

DPKG_MAINTSCRIPT_PACKAGE_REFCOUNT

Defined by **dpkg** on the maintainer script environment to the package reference count, i.e. the number of package instances with a state greater than **not-installed** (since dpkg 1.17.2).

DPKG_MAINTSCRIPT_ARCH

Defined by **dpkg** on the maintainer script environment to the architecture the package got built for (since dpkg 1.15.4).

DPKG_MAINTSCRIPT_NAME

Defined by **dpkg** on the maintainer script environment to the name of the script running, one of **preinst**, **postinst**, **prerm** or **postrm** (since dpkg 1.15.7).

DPKG_MAINTSCRIPT_DEBUG

Defined by **dpkg** on the maintainer script environment to a value (**'0'** or **'1'**) noting whether debugging has been requested (with the **--debug** option) for the maintainer scripts (since dpkg 1.18.4).

FILES*/etc/dpkg/dpkg.cfg.d/[0-9a-zA-Z_-]**

Configuration fragment files (since dpkg 1.15.4).

/etc/dpkg/dpkg.cfg

Configuration file with default options.

*/var/log/dpkg.log*Default log file (see */etc/dpkg/dpkg.cfg* and option **--log**).The other files listed below are in their default directories, see option **--admindir** to see how to change locations of these files.*/var/lib/dpkg/available*

List of available packages.

*/var/lib/dpkg/status*Statuses of available packages. This file contains information about whether a package is marked for removing or not, whether it is installed or not, etc. See section **INFORMATION ABOUT PACKAGES** for more info.The status file is backed up daily in */var/backups*. It can be useful if it's lost or corrupted due to filesystems troubles.The format and contents of a binary package are described in **deb(5)**.**BUGS****--no-act** usually gives less information than might be helpful.**EXAMPLES**To list installed packages related to the editor **vi**(1) (note that **dpkg-query** does not load the *available* file anymore by default, and the **dpkg-query --load-avail** option should be used instead for that):

```
dpkg -l '*vi*'
```

To see the entries in */var/lib/dpkg/available* of two packages:

```
dpkg --print-avail elvis vim | less
```

To search the listing of packages yourself:

```
less /var/lib/dpkg/available
```

To remove an installed elvis package:

```
dpkg -r elvis
```

To install a package, you first need to find it in an archive or CDROM. The *available* file shows that the vim package is in section **editors**:

```
cd /media/cdrom/pool/main/v/vim
dpkg -i vim_4.5-3.deb
```

To make a local copy of the package selection states:

```
dpkg --get-selections> myselections
```

You might transfer this file to another computer, and after having updated the *available* file there with your package manager frontend of choice (see <<https://wiki.debian.org/Teams/Dpkg/FAQ>> for more details), for example:

```
apt-cache dumpavail | dpkg --merge-avail
```

or with dpkg 1.17.6 and earlier:

```
avail=$(mktemp)
apt-cache dumpavail> "$avail"
dpkg --merge-avail "$avail"
rm "$avail"
```

you can install it with:

```
dpkg --clear-selections
dpkg --set-selections <myselections
```

Note that this will not actually install or remove anything, but just set the selection state on the requested packages. You will need some other application to actually download and install the requested packages. For example, run **apt-get dselect-upgrade**.

Ordinarily, you will find that **dselect(1)** provides a more convenient way to modify the package selection states.

ADDITIONAL FUNCTIONALITY

Additional functionality can be gained by installing any of the following packages: **apt**, **aptitude** and **debsums**.

SEE ALSO

aptitude(8), **apt(8)**, **dselect(1)**, **dpkg-deb(1)**, **dpkg-query(1)**, **deb(5)**, **deb-control(5)**, **dpkg.cfg(5)**, and **dpkg-reconfigure(8)**.

AUTHORS

See */usr/share/doc/dpkg/THANKS* for the list of people who have contributed to **dpkg**.

NAME

dpkg.cfg – dpkg configuration file

DESCRIPTION

This file contains default options for dpkg. Each line contains a single option which is exactly the same as a normal command line option for dpkg except for the leading hyphens which are not used here. Quotes surrounding option values are stripped. Comments are allowed by starting a line with a hash sign ('#').

FILES

*/etc/dpkg/dpkg.cfg.d/[0-9a-zA-Z_-]**
/etc/dpkg/dpkg.cfg
~.dpkg.cfg

SEE ALSO

dpkg(1).

NAME

du – estimate file space usage

SYNOPSIS

du [*OPTION*]... [*FILE*]...
du [*OPTION*]... --files0-from=*F*

DESCRIPTION

Summarize disk usage of the set of FILES, recursively for directories.

Mandatory arguments to long options are mandatory for short options too.

-0, --null

end each output line with NUL, not newline

-a, --all

write counts for all files, not just directories

--apparent-size

print apparent sizes, rather than disk usage; although the apparent size is usually smaller, it may be larger due to holes in ('sparse') files, internal fragmentation, indirect blocks, and the like

-B, --block-size=SIZE

scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below

-b, --bytes

equivalent to '--apparent-size --block-size=1'

-c, --total

produce a grand total

-D, --dereference-args

dereference only symlinks that are listed on the command line

-d, --max-depth=N

print the total for a directory (or file, with --all) only if it is N or fewer levels below the command line argument; --max-depth=0 is the same as --summarize

--files0-from=F

summarize disk usage of the NUL-terminated file names specified in file F; if F is -, then read names from standard input

-H

equivalent to --dereference-args (-D)

-h, --human-readable

print sizes in human readable format (e.g., 1K 234M 2G)

--inodes

list inode usage information instead of block usage

-k

like --block-size=1K

-L, --dereference

dereference all symbolic links

-l, --count-links

count sizes many times if hard linked

-m

like --block-size=1M

-P, --no-dereference

don't follow any symbolic links (this is the default)

-S, --separate-dirs

for directories do not include size of subdirectories

--si like **-h**, but use powers of 1000 not 1024

-s, --summarize display only a total for each argument

-t, --threshold=SIZE exclude entries smaller than SIZE if positive, or entries greater than SIZE if negative

--time show time of the last modification of any file in the directory, or any of its subdirectories

--time=WORD show time as WORD instead of modification time: atime, access, use, ctime or status

--time-style=STYLE show times using STYLE, which can be: full-iso, long-iso, iso, or +FORMAT; FORMAT is interpreted like in 'date'

-X, --exclude-from=FILE exclude files that match any pattern in FILE

--exclude=PATTERN exclude files that match PATTERN

-x, --one-file-system skip directories on different file systems

--help display this help and exit

--version output version information and exit

Display values are in units of the first available SIZE from **--block-size**, and the DU_BLOCK_SIZE, BLOCK_SIZE and BLOCKSIZE environment variables. Otherwise, units default to 1024 bytes (or 512 if POSIXLY_CORRECT is set).

The SIZE argument is an integer and optional unit (example: 10K is 10*1024). Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000). Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

PATTERNS

PATTERN is a shell pattern (not a regular expression). The pattern? matches any one character, whereas * matches any string (composed of zero, one or multiple characters). For example, *.o will match any files whose names end in .o. Therefore, the command

```
du --exclude='*.o'
```

will skip all files and subdirectories ending in .o (including the file .o itself).

AUTHOR

Written by Torbjorn Granlund, David MacKenzie, Paul Eggert, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/du>>
or available locally via: info '(coreutils) du invocation'

NAME

e2fsck – check a Linux ext2/ext3/ext4 file system

SYNOPSIS

```
e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ] [ -I|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [ -E extended_options ] [ -z undo_file ] device
```

DESCRIPTION

e2fsck is used to check the ext2/ext3/ext4 family of file systems. For ext3 and ext4 file systems that use a journal, if the system has been shut down uncleanly without any errors, normally, after replaying the committed transactions in the journal, the file system should be marked as clean. Hence, for file systems that use journaling, **e2fsck** will normally replay the journal and exit, unless its superblock indicates that further checking is required.

device is a block device (e.g., */dev/sdc1*) or file containing the file system.

Note that in general it is not safe to run **e2fsck** on mounted file systems. The only exception is if the **-n** option is specified, and **-c**, **-I**, or **-L** options are *not* specified. However, even if it is safe to do so, the results printed by **e2fsck** are not valid if the file system is mounted. If **e2fsck** asks whether or not you should check a file system which is mounted, the only correct answer is “no”. Only experts who really know what they are doing should consider answering this question in any other way.

If **e2fsck** is run in interactive mode (meaning that none of **-y**, **-n**, or **-p** are specified), the program will ask the user to fix each problem found in the file system. A response of ‘y’ will fix the error; ‘n’ will leave the error unfixed; and ‘a’ will fix the problem and all subsequent problems; pressing Enter will proceed with the default response, which is printed before the question mark. Pressing Control-C terminates **e2fsck** immediately.

OPTIONS

- a** This option does the same thing as the **-p** option. It is provided for backwards compatibility only; it is suggested that people use **-p** option whenever possible.
- b** *superblock*
Instead of using the normal superblock, use an alternative superblock specified by *superblock*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system’s blocksize, the number of blocks per group, and features such as **sparse_super**.
Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system’s layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).
If an alternative superblock is specified and the file system is not opened read-only, **e2fsck** will make sure that the primary superblock is updated appropriately upon completion of the file system check.
- B** *blocksize*
Normally, **e2fsck** will search for the superblock at various different block sizes in an attempt to find the appropriate block size. This search can be fooled in some cases. This option forces **e2fsck** to only try locating the superblock at a particular blocksize. If the superblock is not found, **e2fsck** will terminate with a fatal error.
- c** This option causes **e2fsck** to use **badblocks(8)** program to do a read-only scan of the device in order to find any bad blocks. If any bad blocks are found, they are added to the bad block inode to prevent them from being allocated to a file or directory. If this option is specified twice, then the bad block scan will be done using a non-destructive read-write test.
- C** *fd* This option causes **e2fsck** to write completion information to the specified file descriptor so that the progress of the file system check can be monitored. This option is typically used by programs which are running **e2fsck**. If the file descriptor number is negative, then absolute value of the file descriptor will be used, and the progress information will be suppressed initially. It can later be enabled by sending the **e2fsck** process a SIGUSR1 signal. If the file descriptor specified is 0,

e2fsck will print a completion bar as it goes about its business. This requires that e2fsck is running on a video console or terminal.

- d** Print debugging output (useless unless you are debugging **e2fsck**).
- D** Optimize directories in file system. This option causes e2fsck to try to optimize all directories, either by re-indexing them if the file system supports directory indexing, or by sorting and compressing directories for smaller directories, or for file systems using traditional linear directories.

Even without the **-D** option, **e2fsck** may sometimes optimize a few directories --- for example, if directory indexing is enabled and a directory is not indexed and would benefit from being indexed, or if the index structures are corrupted and need to be rebuilt. The **-D** option forces all directories in the file system to be optimized. This can sometimes make them a little smaller and slightly faster to search, but in practice, you should rarely need to use this option.

The **-D** option will detect directory entries with duplicate names in a single directory, which e2fsck normally does not enforce for performance reasons.

-E *extended_options*

Set e2fsck extended options. Extended options are comma separated, and may take an argument using the equals ('=') sign. The following options are supported:

ea_ver=extended_attribute_version

Set the version of the extended attribute blocks which **e2fsck** will require while checking the file system. The version number may be 1 or 2. The default extended attribute version format is 2.

journal_only

Only replay the journal if required, but do not perform any further checks or repairs.

fragcheck

During pass 1, print a detailed report of any discontiguous blocks for files in the file system.

discard

Attempt to discard free blocks and unused inode blocks after the full file system check (discarding blocks is useful on solid state devices and sparse / thin-provisioned storage). Note that discard is done in pass 5 AFTER the file system has been fully checked and only if it does not contain recognizable errors. However there might be cases where **e2fsck** does not fully recognize a problem and hence in this case this option may prevent you from further manual data recovery.

nodiscard

Do not attempt to discard free blocks and unused inode blocks. This option is exactly the opposite of discard option. This is set as default.

no_optimize_extents

Do not offer to optimize the extent tree by eliminating unnecessary width or depth. This can also be enabled in the options section of **/etc/e2fsck.conf**.

optimize_extents

Offer to optimize the extent tree by eliminating unnecessary width or depth. This is the default unless otherwise specified in **/etc/e2fsck.conf**.

inode_count_fullmap

Trade off using memory for speed when checking a file system with a large number of hard-linked files. The amount of memory required is proportional to the number of inodes in the file system. For large file systems, this can be gigabytes of memory. (For example, a 40TB file system with 2.8 billion inodes will consume an additional 5.7 GB memory if this optimization is

enabled.) This optimization can also be enabled in the options section of */etc/e2fsck.conf*.

no_inode_count_fullmap

Disable the **inode_count_fullmap** optimization. This is the default unless otherwise specified in */etc/e2fsck.conf*.

readahead_kb

Use this many KiB of memory to pre-fetch metadata in the hopes of reducing e2fsck runtime. By default, this is set to the size of two block groups' inode tables (typically 4MiB on a regular ext4 file system); if this amount is more than 1/50th of total physical memory, readahead is disabled. Set this to zero to disable readahead entirely.

bmap2extent

Convert block-mapped files to extent-mapped files.

fixes_only

Only fix damaged metadata; do not optimize htree directories or compress extent trees. This option is incompatible with the -D and -E bmap2extent options.

check_encoding

Force verification of encoded filenames in case-insensitive directories. This is the default mode if the file system has the strict flag enabled.

unshare_blocks

If the file system has shared blocks, with the shared blocks read-only feature enabled, then this will unshare all shared blocks and unset the read-only feature bit. If there is not enough free space then the operation will fail. If the file system does not have the read-only feature bit, but has shared blocks anyway, then this option will have no effect. Note when using this option, if there is no free space to clone blocks, there is no prompt to delete files and instead the operation will fail.

Note that **unshare_blocks** implies the "-f" option to ensure that all passes are run. Additionally, if "-n" is also specified, e2fsck will simulate trying to allocate enough space to deduplicate. If this fails, the exit code will be non-zero.

-f Force checking even if the file system seems clean.

-F Flush the file system device's buffer caches before beginning. Only really useful for doing **e2fsck** time trials.

-j *external-journal*

Set the pathname where the external-journal for this file system can be found.

-k When combined with the **-c** option, any existing bad blocks in the bad blocks list are preserved, and any new bad blocks found by running **badblocks(8)** will be added to the existing bad blocks list.

-l *filename*

Add the block numbers listed in the file specified by *filename* to the list of bad blocks. The format of this file is the same as the one generated by the **badblocks(8)** program. Note that the block numbers are based on the blocksize of the file system. Hence, **badblocks(8)** must be given the blocksize of the file system in order to obtain correct results. As a result, it is much simpler and safer to use the **-c** option to **e2fsck**, since it will assure that the correct parameters are passed to the **badblocks** program.

-L *filename*

Set the bad blocks list to be the list of blocks specified by *filename*. (This option is the same as the **-l** option, except the bad blocks list is cleared before the blocks listed in the file are added to

the bad blocks list.)

- n** Open the file system read-only, and assume an answer of ‘no’ to all questions. Allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-p** or **-y** options.
- p** Automatically repair (“preen”) the file system. This option will cause **e2fsck** to automatically fix any file system problems that can be safely fixed without human intervention. If **e2fsck** discovers a problem which may require the system administrator to take additional corrective action, **e2fsck** will print a description of the problem and then exit with the value 4 logically or’ed into the exit code. (See the **EXIT CODE** section.) This option is normally used by the system’s boot scripts. It may not be specified at the same time as the **-n** or **-y** options.
- r** This option does nothing at all; it is provided only for backwards compatibility.
- t** Print timing statistics for **e2fsck**. If this option is used twice, additional timing statistics are printed on a pass by pass basis.
- v** Verbose mode.
- V** Print version information and exit.
- y** Assume an answer of ‘yes’ to all questions; allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-n** or **-p** options.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the **undo_file** argument, the undo file will be written to a file named **e2fsck-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable.

WARNING: The undo file cannot be used to recover from a power or system crash.

EXIT CODE

The exit code returned by **e2fsck** is the sum of the following conditions:

0	– No errors
1	– File system errors corrected
2	– File system errors corrected, system should be rebooted
4	– File system errors left uncorrected
8	– Operational error
16	– Usage or syntax error
32	– E2fsck canceled by user request
128	– Shared library error

SIGNALS

The following signals have the following effect when sent to **e2fsck**.

SIGUSR1

This signal causes **e2fsck** to start displaying a completion bar or emitting progress information. (See discussion of the **-C** option.)

SIGUSR2

This signal causes **e2fsck** to stop displaying a completion bar or emitting progress information.

REPORTING BUGS

Almost any piece of software will have bugs. If you manage to find a file system which causes **e2fsck** to crash, or which **e2fsck** is unable to repair, please report it to the author.

Please include as much information as possible in your bug report. Ideally, include a complete transcript of the **e2fsck** run, so I can see exactly what error messages are displayed. (Make sure the messages printed by **e2fsck** are in English; if your system has been configured so that **e2fsck**’s messages have been translated

into another language, please set the the **LC_ALL** environment variable to **C** so that the transcript of e2fsck's output will be useful to me.) If you have a writable file system where the transcript can be stored, the **script(1)** program is a handy way to save the output of **e2fsck** to a file.

It is also useful to send the output of **dumpe2fs(8)**. If a specific inode or inodes seems to be giving **e2fsck** trouble, try running the **debugfs(8)** command and send the output of the **stat(1u)** command run on the relevant inode(s). If the inode is a directory, the **debugfs** *dump* command will allow you to extract the contents of the directory inode, which can sent to me after being first run through **uuencode(1)**. The most useful data you can send to help reproduce the bug is a compressed raw image dump of the file system, generated using **e2image(8)**. See the **e2image(8)** man page for more details.

Always include the full version string which **e2fsck** displays when it is run, so I know which version you are running.

ENVIRONMENT

E2FSCK_CONFIG

Determines the location of the configuration file (see **e2fsck.conf(5)**).

AUTHOR

This version of **e2fsck** was written by Theodore Ts'o <tytso@mit.edu>.

SEE ALSO

e2fsck.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **debugfs(8)**, **e2image(8)**, **mke2fs(8)**, **tune2fs(8)**

NAME

e2fsck.conf – Configuration file for e2fsck

DESCRIPTION

e2fsck.conf is the configuration file for **e2fsck(8)**. It controls the default behavior of **e2fsck(8)** while it is checking ext2, ext3, or ext4 file systems.

The *e2fsck.conf* file uses an INI-style format. Stanzas, or top-level sections, are delimited by square braces: []. Within each section, each line defines a relation, which assigns tags to values, or to a subsection, which contains further relations or subsections. An example of the INI-style format used by this configuration file follows below:

```
[section1]
tag1 = value_a
tag1 = value_b
tag2 = value_c

[section 2]
tag3 =
    subtag1 = subtag_value_a
    subtag1 = subtag_value_b
    subtag2 = subtag_value_c
}
tag1 = value_d
tag2 = value_e
}
```

Comments are delimited by a semicolon (';') or a hash ('#') character at the beginning of the comment, and are terminated by the end of line character.

Tags and values must be quoted using double quotes if they contain spaces. Within a quoted string, the standard backslash interpretations apply: "\n" (for the newline character), "\t" (for the tab character), "\b" (for the backspace character), and "\\\" (for the backslash character).

The following stanzas are used in the *e2fsck.conf* file. They will be described in more detail in future sections of this document.

[options]

This stanza contains general configuration parameters for **e2fsck**'s behavior.

[defaults]

Contains relations which define the default parameters used by **e2fsck(8)**. In general, these defaults may be overridden by command-line options provided by the user.

[problems]

This stanza allows the administrator to reconfigure how e2fsck handles various file system inconsistencies.

[scratch_files]

This stanza controls when e2fsck will attempt to use scratch files to reduce the need for memory.

THE [options] STANZA

The following relations are defined in the *[options]* stanza.

allow_cancellation

If this relation is set to a boolean value of true, then if the user interrupts e2fsck using ^C, and the file system is not explicitly flagged as containing errors, e2fsck will exit with an exit status of 0 instead of 32. This setting defaults to false.

accept_time_fudge

Unfortunately, due to Windows' unfortunate design decision to configure the hardware clock to tick localtime, instead of the more proper and less error-prone UTC time, many users end up in the situation where the system clock is incorrectly set at the time when e2fsck is run.

Historically this was usually due to some distributions having buggy init scripts and/or installers that didn't correctly detect this case and take appropriate countermeasures. Unfortunately, this is occasionally true even today, usually due to a buggy or misconfigured virtualization manager or the installer not having access to a network time server during the installation process. So by default, we allow the superblock times to be fudged by up to 24 hours. This can be disabled by setting *accept_time_fudge* to the boolean value of false. This setting defaults to true.

broken_system_clock

The **e2fsck**(8) program has some heuristics that assume that the system clock is correct. In addition, many system programs make similar assumptions. For example, the UUID library depends on time not going backwards in order for it to be able to make its guarantees about issuing universally unique ID's. Systems with broken system clocks, are well, broken. However, broken system clocks, particularly in embedded systems, do exist. E2fsck will attempt to use heuristics to determine if the time can not be trusted; and to skip time-based checks if this is true. If this boolean is set to true, then e2fsck will always assume that the system clock can not be trusted.

buggy_init_scripts

This boolean relation is an alias for *accept_time_fudge* for backwards compatibility; it used to be that the behavior defined by *accept_time_fudge* above defaulted to false, and *buggy_init_scripts* would enable superblock time field to be wrong by up to 24 hours. When we changed the default, we also renamed this boolean relation to *accept_time_fudge*.

clear_test_fs_flag

This boolean relation controls whether or not **e2fsck**(8) will offer to clear the test_fs flag if the ext4 file system is available on the system. It defaults to true.

defer_check_on_battery

This boolean relation controls whether or not the interval between file system checks (either based on time or number of mounts) should be doubled if the system is running on battery. This setting defaults to true.

indexed_dir_slack_percentage

When **e2fsck**(8) repacks a indexed directory, reserve the specified percentage of empty space in each leaf nodes so that a few new entries can be added to the directory without splitting leaf nodes, so that the average fill ratio of directories can be maintained at a higher, more efficient level. This relation defaults to 20 percent.

inode_count_fullmap

If this boolean relation is true, trade off using memory for speed when checking a file system with a large number of hard-linked files. The amount of memory required is proportional to the number of inodes in the file system. For large file systems, this can be gigabytes of memory. (For example a 40TB file system with 2.8 billion inodes will consume an additional 5.7 GB memory if this optimization is enabled.) This setting defaults to false.

log_dir If the *log_filename* or *problem_log_filename* relations contains a relative pathname, then the log file will be placed in the directory named by the *log_dir* relation.

log_dir_fallback

This relation contains an alternate directory that will be used if the directory specified by *log_dir* is not available or is not writable.

log_dir_wait

If this boolean relation is true, them if the directories specified by *log_dir* or *log_dir_fallback* are not available or are not yet writable, e2fsck will save the output in a memory buffer, and a child process will periodically test to see if the log directory has become available after the boot sequence has mounted the requested file system for reading/writing. This implements the functionality provided by **logsave**(8) for e2fsck log files.

log_filename

This relation specifies the file name where a copy of e2fsck's output will be written. If certain problem reports are suppressed using the *max_count_problems* relation, (or on a per-problem basis using the *max_count* relation), the full set of problem reports will be written to the log file. The filename may contain various percent-expressions (%D, %T, %N, etc.) which will be expanded so that the file name for the log file can include things like date, time, device name, and other runtime parameters. See the **LOGGING** section for more details.

max_count_problems

This relation specifies the maximum number of problem reports of a particular type will be printed to stdout before further problem reports of that type are squelched. This can be useful if the console is slow (i.e., connected to a serial port) and so a large amount of output could end up delaying the boot process for a long time (potentially hours).

no_optimize_extents

If this boolean relation is true, do not offer to optimize the extent tree by reducing the tree's width or depth. This setting defaults to false.

problem_log_filename

This relation specifies the file name where a log of problem codes found by e2fsck be written. The filename may contain various percent-expressions (%D, %T, %N, etc.) which will be expanded so that the file name for the log file can include things like date, time, device name, and other runtime parameters. See the **LOGGING** section for more details.

readahead_mem_pct

Use this percentage of memory to try to read in metadata blocks ahead of the main e2fsck thread. This should reduce run times, depending on the speed of the underlying storage and the amount of free memory. There is no default, but see **readahead_kb** for more details.

readahead_kb

Use this amount of memory to read in metadata blocks ahead of the main checking thread. Setting this value to zero disables readahead entirely. By default, this is set the size of two block groups' inode tables (typically 4MiB on a regular ext4 file system); if this amount is more than 1/50th of total physical memory, readahead is disabled.

report_features

If this boolean relation is true, e2fsck will print the file system features as part of its verbose reporting (i.e., if the **-v** option is specified)

report_time

If this boolean relation is true, e2fsck will run as if the options **-tt** are always specified. This will cause e2fsck to print timing statistics on a pass by pass basis for full file system checks.

report_verbose

If this boolean relation is true, e2fsck will run as if the option **-v** is always specified. This will cause e2fsck to print some additional information at the end of each full file system check.

THE [defaults] STANZA

The following relations are defined in the *[defaults]* stanza.

undo_dir

This relation specifies the directory where the undo file should be stored. It can be overridden via the **E2FSPROGS_UNDO_DIR** environment variable. If the directory location is set to the value *none*, **e2fsck** will not create an undo file.

THE [problems] STANZA

Each tag in the *[problems]* stanza names a problem code specified with a leading "0x" followed by six hex digits. The value of the tag is a subsection where the relations in that subsection override the default treatment of that particular problem code.

Note that inappropriate settings in this stanza may cause **e2fsck** to behave incorrectly, or even crash. Most system administrators should not be making changes to this section without referring to source code.

Within each problem code's subsection, the following tags may be used:

description

This relation allows the message which is printed when this file system inconsistency is detected to be overridden.

preen_ok

This boolean relation overrides the default behavior controlling whether this file system problem should be automatically fixed when **e2fsck** is running in preen mode.

max_count

This integer relation overrides the *max_count_problems* parameter (set in the options section) for this particular problem.

no_ok This boolean relation overrides the default behavior determining whether or not the file system will be marked as inconsistent if the user declines to fix the reported problem.

no_default

This boolean relation overrides whether the default answer for this problem (or question) should be "no".

preen_nomessage

This boolean relation overrides the default behavior controlling whether or not the description for this file system problem should be suppressed when **e2fsck** is running in preen mode.

no_nomsg

This boolean relation overrides the default behavior controlling whether or not the description for this file system problem should be suppressed when a problem forced not to be fixed, either because **e2fsck** is run with the **-n** option or because the *force_no* flag has been set for the problem.

force_no

This boolean option, if set to true, forces a problem to never be fixed. That is, it will be as if the user problem responds 'no' to the question of 'should this problem be fixed?'. The *force_no* option even overrides the **-y** option given on the command-line (just for the specific problem, of course).

not_a_fix

This boolean option, if set to true, marks the problem as one where if the user gives permission to make the requested change, it does not mean that the file system had a problem which has since been fixed. This is used for requests to optimize the file system's data structure, such as pruning an extent tree.

THE [scratch_files] STANZA

The following relations are defined in the *[scratch_files]* stanza.

directory

If the directory named by this relation exists and is writeable, then e2fsck will attempt to use this directory to store scratch files instead of using in-memory data structures.

numdirs_threshold

If this relation is set, then in-memory data structures will be used if the number of directories in the file system are fewer than amount specified.

dirinfo This relation controls whether or not the scratch file directory is used instead of an in-memory data structure for directory information. It defaults to true.

icount This relation controls whether or not the scratch file directory is used instead of an in-memory data structure when tracking inode counts. It defaults to true.

LOGGING

E2fsck has the facility to save the information from an e2fsck run in a directory so that a system administrator can review its output at their leisure. This allows information captured during the automatic e2fsck preen run, as well as a manually started e2fsck run, to be saved for posterity. This facility is controlled by

the *log_filename*, *log_dir*, *log_dir_fallback*, and *log_dir_wait* relations in the *[options]* stanza.

The filename in *log_filename* may contain the following percent-expressions that will be expanded as follows.

%d	The current day of the month
%D	The current date; this is equivalent of %Y%m%d
%h	The hostname of the system.
%H	The current hour in 24-hour format (00..23)
%m	The current month as a two-digit number (01..12)
%M	The current minute (00..59)
%N	The name of the block device containing the file system, with any directory pathname stripped off.
%p	The pid of the e2fsck process
%s	The current time expressed as the number of seconds since 1970-01-01 00:00:00 UTC
%S	The current second (00..59)
%T	The current time; this is equivalent of %H%M%S
%u	The name of the user running e2fsck.
%U	This percent expression does not expand to anything, but it signals that any following date or time expressions should be expressed in UTC time instead of the local timezone.
%y	The last two digits of the current year (00..99)
%Y	The current year (i.e., 2012).

EXAMPLES

The following recipe will prevent e2fsck from aborting during the boot process when a file system contains orphaned files. (Of course, this is not always a good idea, since critical files that are needed for the security of the system could potentially end up in lost+found, and starting the system without first having a system administrator check things out may be dangerous.)

```
[problems]
0x040002 = {
    preen_ok = true
    description = "@u @i %i. "
}
```

The following recipe will cause an e2fsck logfile to be written to the directory /var/log/e2fsck, with a filename that contains the device name, the hostname of the system, the date, and time: e.g., "e2fsck-sda3.server.INFO.20120314-112142". If the directory containing /var/log is located on the root file system which is initially mounted read-only, then the output will be saved in memory and written out once the root file system has been remounted read/write. To avoid too much detail from being written to the serial console (which could potentially slow down the boot sequence), only print no more than 16 instances of each type of file system corruption.

```
[options]
max_count_problems = 16
log_dir = /var/log/e2fsck
log_filename = e2fsck-%N.%h.INFO.%D-%T
log_dir_wait = true
```

FILES

/etc/e2fsck.conf

The configuration file for **e2fsck(8)**.

SEE ALSO**e2fsck(8)**

emacs(1) - Linux man page

Name

emacs - GNU project Emacs

Synopsis

emacs [*command-line switches*] [*files ...*]

Description

GNU Emacs is a version of *Emacs*, written by the author of the original (PDP-10) *Emacs*, Richard Stallman.

The primary documentation of *GNU Emacs* is in the *GNU Emacs Manual*, which you can read using Info, either from *Emacs* or as a standalone program. Please look there for complete and up-to-date documentation. This man page is updated only when someone volunteers to do so; the *Emacs* maintainers' priority goal is to minimize the amount of time this man page takes away from other more useful projects.

The user functionality of *GNU Emacs* encompasses everything other *Emacs* editors do, and it is easily extensible since its editing commands are written in Lisp.

Emacs has an extensive interactive help facility, but the facility assumes that you know how to manipulate *Emacs* windows and buffers. CTRL-h or F1 enters the Help facility. Help Tutorial (CTRL-h t) starts an interactive tutorial which can teach beginners the fundamentals of *Emacs* in a few minutes. Help Apropos (CTRL-h a) helps you find a command given its functionality, Help Character (CTRL-h c) describes a given character's effect, and Help Function (CTRL-h f) describes a given Lisp function specified by name.

Emacs's Undo can undo several steps of modification to your buffers, so it is easy to recover from editing mistakes.

GNU Emacs's many special packages handle mail reading (RMail) and sending (Mail), outline editing (Outline), compiling (Compile), running subshells within *Emacs* windows (Shell), running a Lisp read-eval-print loop (Lisp-Interaction-Mode), automated psychotherapy (Doctor), and much more.

There is an extensive reference manual, but users of other *Emacs*es should have little trouble adapting even without a copy. Users new to *Emacs* will be able to use basic features fairly rapidly by studying the tutorial and using the self-documentation features.

Emacs Options

The following options are of general interest:

file

Edit *file*.

--file*file*, **--find-file** *file*, **--visit** *file*

The same as specifying *file* directly as an argument.

+*number*

Go to the line specified by *number* (do not insert a space between the "+" sign and the number). This applies only to the next file specified.

+*line:column*

Go to the specified *line* and *column*.

-q, **--no-init-file**

Do not load an init file.

--no-site-file

Do not load the site-wide startup file.

--no-desktop

Do not load a saved desktop.

-nI, **--no-shared-memory**

Do not use shared memory.

-Q, **--quick**

Equivalent to "-q --no-site-file --no-splash".

--no-splash

Do not display a splash screen during start-up.

--debug-init

Enable *Emacs* Lisp debugger during the processing of the user init file *~/.emacs*. This is useful for debugging problems in the init file.

-u *user*, **--user** *user*

Load *user*'s init file.

-t *file*, **--terminal** *file*

Use specified *file* as the terminal instead of using stdin/stdout. This must be the first argument specified in the command line.

--multibyte, **--no-unibyte**

Enable multibyte mode (enabled by default).

--unibyte, **--no-multibyte**

Enable unibyte mode.

--version

Display *Emacs* version information and exit.

--help

Display this help and exit.

The following options are lisp-oriented (these options are processed in the order encountered):

-f *function*, **--funcall***function*

Execute the lisp function *function*.

-l *file*, **--load** *file*

Load the lisp code in the file *file*.

--eval expr, --execute expr

Evaluate the Lisp expression *expr*.

The following options are useful when running *Emacs* as a batch editor:

--batch

Edit in batch mode. The editor will send messages to stderr. This option must be the first in the argument list. You must use -l and -f options to specify files to execute and functions to call.

--script file

Run *file* as an Emacs Lisp script.

--insert file

Insert contents of *file* into the current buffer.

--kill

Exit *Emacs* while in batch mode.

-L dir, --directory dir

Add *dir* to the list of directories *Emacs* searches for Lisp files.

Using Emacs with X

Emacs has been tailored to work well with the X window system. If you run *Emacs* from under X windows, it will create its own X window to display in. You will probably want to start the editor as a background process so that you can continue using your original window.

Emacs can be started with the following X switches:

--name name

Specify the name which should be assigned to the initial *Emacs* window.

This controls looking up X resources as well as the window title.

-T name, --title name

Specify the title for the initial X window.

-r, -rv, --reverse-video

Display the *Emacs* window in reverse video.

-fn font, --font font

Set the *Emacs* window's font to that specified by *font*. You will find the various X fonts in the */usr/lib/X11/fonts* directory. Note that *Emacs* will only accept fixed width fonts. Under the X11 Release 4 font-naming conventions, any font with the value "m" or "c" in the eleventh field of the font name is a fixed width font. Furthermore, fonts whose names are of the form *widthxheight* are generally fixed width, as is the font *fixed*. See [xlsfonts\(1\)](#) for more information.

When you specify a font, be sure to put a space between the switch and the font name.

--xrm *resources*

Set additional X resources.

--color, --color=mode

Override color mode for character terminals; *mode* defaults to 'auto', and can also be 'never', 'auto', 'always', or a mode name like 'ansi8'.

-bw *pixels*, **--border-width** *pixels*

Set the *Emacs* window's border width to the number of pixels specified by *pixels*. Defaults to one pixel on each side of the window.

-ib *pixels*, **--internal-border** *pixels*

Set the window's internal border width to the number of pixels specified by *pixels*. Defaults to one pixel of padding on each side of the window.

-g *geometry*, **--geometry** *geometry*

Set the *Emacs* window's width, height, and position as specified. The geometry specification is in the standard X format; see [X\(7\)](#) for more information. The width and height are specified in characters; the default is 80 by 24. See the *Emacs* manual, section "Options for Window Size and Position", for information on how window sizes interact with selecting or deselecting the tool bar and menu bar.

-lsp *pixels*, **--line-spacing** *pixels*

Additional space to put between lines.

-vb, --vertical-scroll-bars

Enable vertical scrollbars.

-fh, --fullheight

Make the first frame as high as the screen.

-fs, --fullscreen

Make the first frame fullscreen.

-fw, --fullwidth

Make the first frame as wide as the screen.

-fg *color*, **--foreground-color** *color*

On color displays, set the color of the text.

Use the command *M-x list-colors-display* for a list of valid color names.

-bg *color*, **--background-color** *color*

On color displays, set the color of the window's background.

-bd *color*, **--border-color** *color*

On color displays, set the color of the window's border.

-cr *color*, **--cursor-color** *color*

On color displays, set the color of the window's text cursor.

-ms *color*, **--mouse-color** *color*

On color displays, set the color of the window's mouse cursor.

-d *displayname*, **--display** *displayname*

Create the *Emacs* window on the display specified by *displayname*. Must be the first option specified in the command line.

-nbi, --no-bitmap-icon

Do not use picture of gnu for Emacs icon.

--iconic

Start *Emacs* in iconified state.

-nbc, --no-blinking-cursor

Disable blinking cursor.

-nw, --no-window-system

Tell *Emacs* not to use its special interface to X. If you use this switch when invoking *Emacs* from an [*xterm*\(1\)](#) window, display is done in that window.

-D, --basic-display

This option disables many display features; use it for debugging *Emacs*.

You can set X default values for your *Emacs* windows in your *.Xresources* file (see [*xrdb*\(1\)](#)). Use the following format:

emacs.keyword:value

where *value* specifies the default value of *keyword*. *Emacs* lets you set default values for the following keywords:

background (class Background)

For color displays, sets the window's background color.

bitmapIcon (class BitmapIcon)

If **bitmapIcon**'s value is set to *on*, the window will iconify into the "kitchen sink."

borderColor (class BorderColor)

For color displays, sets the color of the window's border.

borderWidth (class BorderWidth)

Sets the window's border width in pixels.

cursorColor (class Foreground)

For color displays, sets the color of the window's text cursor.

cursorBlink (class CursorBlink)

Specifies whether to make the cursor blink. The default is *on*. Use *off* or *false* to turn cursor blinking off.

font (class Font)

Sets the window's text font.

foreground (class Foreground)

For color displays, sets the window's text color.

fullscreen (class Fullscreen)

The desired fullscreen size. The value can be one of *fullboth*, *fullwidth*, or *fullheight*, which correspond to the command-line options '*-fs*', '*-fw*', and '*-fh*', respectively. Note that this applies to the initial frame only.

geometry (class Geometry)

Sets the geometry of the *Emacs* window (as described above).

iconName (class **Title**)

Sets the icon name for the *Emacs* window icon.

internalBorder (class **BorderWidth**)

Sets the window's internal border width in pixels.

lineSpacing (class **LineSpacing**)

Additional space ("leading") between lines, in pixels.

menuBar (class **MenuBar**)

Gives frames menu bars if *on*; don't have menu bars if *off*. See the Emacs manual, sections "Lucid Resources" and "LessTif Resources", for how to control the appearance of the menu bar if you have one.

minibuffer (class **Minibuffer**)

If *none*, don't make a minibuffer in this frame. It will use a separate minibuffer frame instead.

paneFont (class **Font**)

Font name for menu pane titles, in non-toolkit versions of *Emacs*.

pointerColor (class **Foreground**)

For color displays, sets the color of the window's mouse cursor.

privateColormap (class **PrivateColormap**)

If *on*, use a private color map, in the case where the "default visual" of class **PseudoColor** and **Emacs** is using it.

reverseVideo (class **ReverseVideo**)

If **reverseVideo**'s value is set to *on*, the window will be displayed in reverse video.

screenGamma (class **ScreenGamma**)

Gamma correction for colors, equivalent to the frame parameter 'screen-gamma'.

scrollBarWidth (class **ScrollBarWidth**)

The scroll bar width in pixels, equivalent to the frame parameter 'scroll-bar-width'.

selectionFont (class **SelectionFont**)

Font name for pop-up menu items, in non-toolkit versions of *Emacs*. (For toolkit versions, see the Emacs manual, sections "Lucid Resources" and "LessTif Resources".)

selectionTimeout (class **SelectionTimeout**)

Number of milliseconds to wait for a selection reply. A value of 0 means wait as long as necessary.

synchronous (class **Synchronous**)

Run Emacs in synchronous mode if *on*. Synchronous mode is useful for debugging X problems.

title (class **Title**)

Sets the title of the *Emacs* window.

toolBar (class **ToolBar**)

Number of lines to reserve for the tool bar.

useXIM (class **UseXIM**)

Turns off use of X input methods (XIM) if *false* or *off*.

verticalScrollBars (class **ScrollBars**)

Gives frames scroll bars if *on*; suppresses scroll bars if *off*.

visualClass (class **VisualClass**)

Specify the "visual" that X should use. This tells X how to handle colors. The value should start with one of *TrueColor*, *PseudoColor*, *DirectColor*, *StaticColor*, *GrayScale*, and *StaticGray*, followed by *-depth*, where *depth* is the number of color planes.

If you try to set color values while using a black and white display, the window's characteristics will default as follows: the foreground color will be set to black, the background color will be set to white, the border color will be set to grey, and the text and mouse cursors will be set to black.

Using the Mouse

The following lists some of the mouse button bindings for the *Emacs* window under X11.

MOUSE BUTTON	FUNCTION
left	Set point.
middle	Paste text.
right	Cut text into X cut buffer.
SHIFT- middle	Cut text into X cut buffer.
SHIFT-right	Paste text.
CTRL-middle	Cut text into X cut buffer and kill it.
CTRL-right	Select this window, then split it into two windows. Same as typing CTRL-x 2.
CTRL- SHIFT-left	X buffer menu -- hold the buttons and keys down, wait for menu to appear, select buffer, and release. Move mouse out of menu and release to cancel.
CTRL- SHIFT- middle	X help menu -- pop up index card menu for Emacs help.
CTRL- SHIFT-right	Select window with mouse, and delete all other windows. Same as typing CTRL-x 1.

Manuals

You can order printed copies of the GNU Emacs Manual from the Free Software Foundation, which develops GNU software. See the file ORDERS for ordering information.

Your local Emacs maintainer might also have copies available. As with all software and publications from FSF, everyone is permitted to make and distribute copies of the Emacs manual. The TeX source to the manual is also included in the Emacs source distribution.

Files

/usr/local/share/info -- files for the Info documentation browser. The complete text of the Emacs reference manual is included in a convenient tree structured form. Also includes the Emacs Lisp Reference Manual, useful to anyone wishing to write programs in the Emacs Lisp extension language.

/usr/local/share/emacs/\$VERSION/lisp -- Lisp source files and compiled files that define most editing commands. Some are preloaded; others are autoloaded from this directory when used.

/usr/local/libexec/emacs/\$VERSION/\$ARCH -- various programs that are used with GNU Emacs.

/usr/local/share/emacs/\$VERSION/etc -- various files of information.

/usr/local/share/emacs/\$VERSION/etc/DOC.* -- contains the documentation strings for the Lisp primitives and preloaded Lisp functions of GNU Emacs. They are stored here to reduce the size of Emacs proper.

/usr/local/share/emacs/\$VERSION/etc/SERVICE lists people offering various services to assist users of GNU Emacs, including education, troubleshooting, porting and customization.

Bugs

There is a mailing list, bug-gnu-emacs@gnu.org, for reporting Emacs bugs and fixes. But before reporting something as a bug, please try to be sure that it really is a bug, not a misunderstanding or a deliberate feature. We ask you to read the section "Reporting Emacs Bugs" near the end of the reference manual (or Info system) for hints on how and when to report bugs. Also, include the version number of the Emacs you are running in every bug report that you send in. Bugs tend actually to be fixed if they can be isolated, so it is in your interest to report them in such a way that they can be easily reproduced.

Do not expect a personal answer to a bug report. The purpose of reporting bugs is to get them fixed for everyone in the next release, if possible. For personal assistance, look in the SERVICE file (see above) for a list of people who offer it.

Please do not send anything but bug reports to this mailing list. For more information about Emacs mailing lists, see the file

/usr/local/share/emacs/\$VERSION/etc/MAILINGLISTS.

Unrestrictions

Emacs is free; anyone may redistribute copies of *Emacs* to anyone under the terms stated in the *Emacs* General Public License, a copy of which accompanies each copy of *Emacs* and which also appears in the reference manual.

Copies of *Emacs* may sometimes be received packaged with distributions of Unix systems, but it is never included in the scope of any license covering those systems. Such inclusion violates the terms on which distribution is permitted. In fact, the primary purpose of the General Public License is to prohibit anyone from attaching any other restrictions to redistribution of *Emacs*.

Richard Stallman encourages you to improve and extend *Emacs*, and urges that you contribute your extensions to the GNU library. Eventually GNU (Gnu's Not Unix) will be a complete replacement for Unix. Everyone will be free to use, copy, study and change the GNU system.

See Also

[emacsclient](#)(1), ***[etags](#)***(1), ***[x\(7\)](#)***, ***[xlsfonts](#)***(1), ***[xterm](#)***(1), ***[xrdb](#)***(1)

Authors

Emacs was written by Richard Stallman and the Free Software Foundation. Joachim Martillo and Robert Krawitz added the X features.

Copying

Copyright © 1995, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009
Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this document under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this document into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Free Software Foundation.

Referenced By

ash(1), **bgltags**(1), **byobu-ctrl-a**(1), **cdecl**(1), **ci**(1), **clisp**(1), **csh**(1), **dash**(1),
ebrowse(1), **eplain**(1), **gp**(1), **gqview**(1), **groff_out**(5), **hebcal**(1), **info**(5), **ksh93**(1),
lacheck(1), **lbdbq**(1), **ldp**(7), **mksh**(1), **perldl**(1), **powwow**(6), **pure**(1), **roff**(7),
sh(1), **texinfo**(5), **unzip**(1), **xemacs**(1), **zile**(1)

NAME

env – run a program in a modified environment

SYNOPSIS

env [*OPTION*]... [-] [*NAME=VALUE*]... [*COMMAND* [*ARG*]...]

DESCRIPTION

Set each NAME to VALUE in the environment and run COMMAND.

Mandatory arguments to long options are mandatory for short options too.

-i, --ignore-environment

start with an empty environment

-0, --null

end each output line with NUL, not newline

-u, --unset=NAME

remove variable from the environment

-C, --chdir=DIR

change working directory to DIR

-S, --split-string=S

process and split S into separate arguments; used to pass multiple arguments on shebang lines

--block-signal[=SIG]

block delivery of SIG signal(s) to COMMAND

--default-signal[=SIG]

reset handling of SIG signal(s) to the default

--ignore-signal[=SIG]

set handling of SIG signals(s) to do nothing

--list-signal-handling

list non default signal handling to stderr

-v, --debug

print verbose information for each processing step

--help

display this help and exit

--version

output version information and exit

A mere – implies **-i**. If no COMMAND, print the resulting environment.

SIG may be a signal name like 'PIPE', or a signal number like '13'. Without SIG, all known signals are included. Multiple signals can be comma-separated.

OPTIONS**-S/--split-string usage in scripts**

The **-S** option allows specifying multiple parameters in a script. Running a script named **1.pl** containing the following first line:

```
#!/usr/bin/env -S perl -w -T
```

```
...
```

Will execute **perl -w -T 1.pl**.

Without the '**-S**' parameter the script will likely fail with:

```
/usr/bin/env: 'perl -w -T': No such file or directory
```

See the full documentation for more details.

--default-signal[=SIG] usage

This option allows setting a signal handler to its default action, which is not possible using the traditional shell trap command. The following example ensures that seq will be terminated by SIGPIPE no matter how this signal is being handled in the process invoking the command.

```
sh -c 'env --default-signal=PIPE seq inf | head -n1'
```

NOTES

POSIX's exec(2) pages says:

"many existing applications wrongly assume that they start with certain signals set to the default action and/or unblocked.... Therefore, it is best not to block or ignore signals across execs without explicit reason to do so, and especially not to block signals across execs of arbitrary (not closely cooperating) programs."

AUTHOR

Written by Richard Mlynarik, David MacKenzie, and Assaf Gordon.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`sigaction(2)`, `sigprocmask(2)`, `signal(7)`

Full documentation <<https://www.gnu.org/software/coreutils/env>>

or available locally via: `info '(coreutils) env invocation'`

NAME

environ – user environment

SYNOPSIS

```
extern char **environ;
```

DESCRIPTION

The variable *environ* points to an array of pointers to strings called the "environment". The last pointer in this array has the value NULL. This array of strings is made available to the process by the **execve(2)** call when a new program is started. When a child process is created via **fork(2)**, it inherits a *copy* of its parent's environment.

By convention, the strings in *environ* have the form "*name=value*". The name is case-sensitive and may not contain the character "="". The value can be anything that can be represented as a string. The name and the value may not contain an embedded null byte ('\0'), since this is assumed to terminate the string.

Environment variables may be placed in the shell's environment by the *export* command in **sh(1)**, or by the *setenv* command if you use **csh(1)**.

The initial environment of the shell is populated in various ways, such as definitions from */etc/environment* that are processed by **pam_env(8)** for all users at login time (on systems that employ **pam(8)**). In addition, various shell initialization scripts, such as the system-wide */etc/profile* script and per-user initializations script may include commands that add variables to the shell's environment; see the manual page of your preferred shell for details.

Bourne-style shells support the syntax

```
NAME=value command
```

to create an environment variable definition only in the scope of the process that executes *command*. Multiple variable definitions, separated by white space, may precede *command*.

Arguments may also be placed in the environment at the point of an **exec(3)**. A C program can manipulate its environment using the functions **getenv(3)**, **putenv(3)**, **setenv(3)**, and **unsetenv(3)**.

What follows is a list of environment variables typically seen on a system. This list is incomplete and includes only common variables seen by average users in their day-to-day routine. Environment variables specific to a particular program or library function are documented in the ENVIRONMENT section of the appropriate manual page.

USER The name of the logged-in user (used by some BSD-derived programs). Set at login time, see section NOTES below.

LOGNAME

The name of the logged-in user (used by some System-V derived programs). Set at login time, see section NOTES below.

HOME

A user's login directory. Set at login time, see section NOTES below.

LANG The name of a locale to use for locale categories when not overridden by **LC_ALL** or more specific environment variables such as **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME** (see **locale(7)** for further details of the **LC_*** environment variables).

PATH The sequence of directory prefixes that **sh(1)** and many other programs employ when searching for an executable file that is specified as a simple filename (i.a., a pathname that contains no slashes). The prefixes are separated by colons (:). The list of prefixes is searched from beginning to end, by checking the pathname formed by concatenating a prefix, a slash, and the filename, until a file with execute permission is found.

As a legacy feature, a zero-length prefix (specified as two adjacent colons, or an initial or terminating colon) is interpreted to mean the current working directory. However, use of this feature is deprecated, and POSIX notes that a conforming application shall use an explicit pathname (e.g., .)

to specify the current working directory.

Analogously to **PATH**, one has **CDPATH** used by some shells to find the target of a change directory command, **MANPATH** used by **man(1)** to find manual pages, and so on.

PWD Absolute path to the current working directory; required to be partially canonical (no . or .. components).

SHELL

The absolute pathname of the user's login shell. Set at login time, see section NOTES below.

TERM The terminal type for which output is to be prepared.

PAGER

The user's preferred utility to display text files. Any string acceptable as a command_string operand to the *sh -c* command shall be valid. If **PAGER** is null or is not set, then applications that launch a pager will default to a program such as **less(1)** or **more(1)**.

EDITOR/VISUAL

The user's preferred utility to edit text files. Any string acceptable as a command_string operand to the *sh -c* command shall be valid.

Note that the behavior of many programs and library routines is influenced by the presence or value of certain environment variables. Examples include the following:

- The variables **LANG**, **LANGUAGE**, **NLSPATH**, **LOCPATH**, **LC_ALL**, **LC_MESSAGES**, and so on influence locale handling; see **catopen(3)**, **gettext(3)**, and **locale(7)**.
- **TMPDIR** influences the path prefix of names created by **tempnam(3)** and other routines, and the temporary directory used by **sort(1)** and other programs.
- **LD_LIBRARY_PATH**, **LD_PRELOAD**, and other **LD_*** variables influence the behavior of the dynamic loader/linker. See **alsold.so(8)**.
- **POSIXLY_CORRECT** makes certain programs and library routines follow the prescriptions of POSIX.
- The behavior of **malloc(3)** is influenced by **MALLOC_*** variables.
- The variable **HOSTALIASES** gives the name of a file containing aliases to be used with **gethostbyname(3)**.
- **TZ** and **TZDIR** give timezone information used by **tzset(3)** and through that by functions like **ctime(3)**, **localtime(3)**, **mktime(3)**, **strftime(3)**. See also **tzselect(8)**.
- **TERMCAP** gives information on how to address a given terminal (or gives the name of a file containing such information).
- **COLUMNS** and **LINES** tell applications about the window size, possibly overriding the actual size.
- **PRINTER** or **LPDEST** may specify the desired printer to use. See **lpr(1)**.

NOTES

Historically and by standard, *environ* must be declared in the user program. However, as a (nonstandard) programmer convenience, *environ* is declared in the header file <*unistd.h*> if the **_GNU_SOURCE** feature test macro is defined (see **feature_test_macros(7)**).

The **prctl(2)** **PR_SET_MM_ENV_START** and **PR_SET_MM_ENV_END** operations can be used to control the location of the process's environment.

The **HOME**, **LOGNAME**, **SHELL**, and **USER** variables are set when the user is changed via a session management interface, typically by a program such as **login(1)** from a user database (such as **passwd(5)**). (Switching to the root user using **su(1)** may result in a mixed environment where **LOGNAME** and **USER** are retained from old user; see the **su(1)** manual page.)

BUGS

Clearly there is a security risk here. Many a system command has been tricked into mischief by a user who specified unusual values for **IFS** or **LD_LIBRARY_PATH**.

There is also the risk of name space pollution. Programs like *make* and *autoconf* allow overriding of default utility names from the environment with similarly named variables in all caps. Thus one uses **CC** to select the desired C compiler (and similarly **MAKE**, **AR**, **AS**, **FC**, **LD**, **LEX**, **RM**, **YACC**, etc.). However, in some traditional uses such an environment variable gives options for the program instead of a pathname. Thus, one has **MORE** and **LESS**. Such usage is considered mistaken, and to be avoided in new programs.

SEE ALSO

bash(1), **csh(1)**, **env(1)**, **login(1)**, **printenv(1)**, **sh(1)**, **su(1)**, **tcsesh(1)**, **execve(2)**, **clearenv(3)**, **exec(3)**, **getenv(3)**, **putenv(3)**, **setenv(3)**, **unsetenv(3)**, **locale(7)**, **ld.so(8)**, **pam_env(8)**

NAME

environment.d – Definition of user service environment

SYNOPSIS

```
~/.config/environment.d/*.conf
/etc/environment.d/*.conf
/run/environment.d/*.conf
/usr/lib/environment.d/*.conf
/etc/environment
```

DESCRIPTION

Configuration files in the environment.d/ directories contain lists of environment variable assignments for services started by the systemd user instance. **systemd-environment-d-generator(8)** parses them and updates the environment exported by the systemd user instance. See below for an discussion of which processes inherit those variables.

It is recommended to use numerical prefixes for file names to simplify ordering.

For backwards compatibility, a symlink to /etc/environment is installed, so this file is also parsed.

CONFIGURATION DIRECTORIES AND PRECEDENCE

Configuration files are read from directories in /etc/, /run/, /usr/local/lib/, and /lib/, in order of precedence, as listed in the SYNOPSIS section above. Files must have the ".conf" extension. Files in /etc/ override files with the same name in /run/, /usr/local/lib/, and /lib/. Files in /run/ override files with the same name under /usr/.

All configuration files are sorted by their filename in lexicographic order, regardless of which of the directories they reside in. If multiple files specify the same option, the entry in the file with the lexicographically latest name will take precedence. Thus, the configuration in a certain file may either be replaced completely (by placing a file with the same name in a directory with higher priority), or individual settings might be changed (by specifying additional settings in a file with a different name that is ordered later).

Packages should install their configuration files in /usr/lib/ (distribution packages) or /usr/local/lib/ (local installs). Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. It is recommended to prefix all filenames with a two-digit number and a dash, to simplify the ordering of the files.

If the administrator wants to disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file. If the vendor configuration file is included in the initrd image, the image has to be regenerated.

CONFIGURATION FORMAT

The configuration files contain a list of "KEY=VALUE" environment variable assignments, separated by newlines. The right hand side of these assignments may reference previously defined environment variables, using the "\${OTHER_KEY}" and "\$OTHER_KEY" format. It is also possible to use "\${FOO:-DEFAULT_VALUE}" to expand in the same way as "\${FOO}" unless the expansion would be empty, in which case it expands to DEFAULT_VALUE, and use "\${FOO:+ALTERNATE_VALUE}" to expand to ALTERNATE_VALUE as long as "\${FOO}" would have expanded to a non-empty value. No other elements of shell syntax are supported.

Each KEY must be a valid variable name. Empty lines and lines beginning with the comment character "#" are ignored.

Example**Example 1. Setup environment to allow access to a program installed in /opt/foo**

/etc/environment.d/60-foo.conf:

```
FOO_DEBUG=force-software-gl,log-verbose
PATH=/opt/foo/bin:$PATH
LD_LIBRARY_PATH=/opt/foo/lib${LD_LIBRARY_PATH:+:$LD_LIBRARY_PATH}
XDG_DATA_DIRS=/opt/foo/share:${XDG_DATA_DIRS:-/usr/local/share:/usr/share/}
```

APPLICABILITY

Environment variables exported by the user manager (**systemd --user** instance started in the `user@uid.service` system service) apply to any services started by that manager. In particular, this may include services which run user shells. For example in the GNOME environment, the graphical terminal emulator runs as the `gnome-terminal-server.service` user unit, which in turn runs the user shell, so that shell will inherit environment variables exported by the user manager. For other instances of the shell, not launched by the user manager, the environment they inherit is defined by the program that starts them. Hint: in general, **systemd.service(5)** units contain programs launched by systemd, and **systemd.scope(5)** units contain programs launched by something else.

Specifically, for ssh logins, the **sshd(8)** service builds an environment that is a combination of variables forwarded from the remote system and defined by **sshd**, see the discussion in **ssh(1)**. A graphical display session will have an analogous mechanism to define the environment. Note that some managers query the systemd user instance for the exported environment and inject this configuration into programs they start, using **systemctl show-environment** or the underlying D-Bus call.

SEE ALSO

systemd(1), **systemd-environment-d-generator(8)**, **systemd.environment-generator(7)**

NAME

ethtool – query or control network driver and hardware settings

SYNOPSIS

```

ethtool devname
ethtool -h|--help
ethtool --version
ethtool [--debug N] args
ethtool [--json] args
ethtool [-I | --include-statistics] args
ethtool --monitor [ command ] [ devname ]
ethtool -a|--show-pause devname
ethtool -A|--pause devname [autoneg on|off] [rx on|off] [tx on|off]
ethtool -c|--show-coalesce devname
ethtool -C|--coalesce devname [adaptive-rx on|off] [adaptive-tx on|off] [rx-usecs N] [rx-frames N]
   [rx-usecs-irq N] [rx-frames-irq N] [tx-usecs N] [tx-frames N] [tx-usecs-irq N]
   [tx-frames-irq N] [stats-block-usecs N] [pkt-rate-low N] [rx-usecs-low N]
   [rx-frames-low N] [tx-usecs-low N] [tx-frames-low N] [pkt-rate-high N]
   [rx-usecs-high N] [rx-frames-high N] [tx-usecs-high N] [tx-frames-high N]
   [sample-interval N] [cqe-mode-rx on|off] [cqe-mode-tx on|off]
ethtool -g|--show-ring devname
ethtool -G|--set-ring devname [rx N] [rx-mini N] [rx-jumbo N] [tx N]
ethtool -i|--driver devname
ethtool -d|--register-dump devname [raw on|off] [hex on|off] [file name]
ethtool -e|--eeprom-dump devname [raw on|off] [offset N] [length N]
ethtool -E|--change-eeprom devname [magic N] [offset N] [length N] [value N]
ethtool -k|--show-features|--show-offload devname
ethtool -K|--features|--offload devname feature on|off ...
ethtool -p|--identify devname [N]
ethtool -P|--show-permaddr devname
ethtool -r|--negotiate devname
ethtool -S|--statistics devname [--all-groups|--groups [eth-phy] [eth-mac] [eth-ctrl] ]
ethtool --phy-statistics devname
ethtool -t|--test devname [offline|online|external_lb]
ethtool -s devname [speed N] [lanes N] [duplex half|full] [port tp|au|bnc|mii] [mdix auto|on|off]
   [autoneg on|off] [advertise N|M] [advertise mode on|off ...] [phyad N]
   [xcvr internal|external] [wol N|M] [wol p|u|m|b|a|g|s|f|d...] [sopass xx:yy:zz:aa:bb:cc]
   [master-slave preferred-master|preferred-slave|forced-master|forced-slave] [msglvl N|M] [msglvl type on|off ...]
ethtool -n|--show-nfc|--show-ntuple devname
   [rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6 | rule N]
ethtool -N|--config-nfc|--config-ntuple devname
   rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6 m|v|t|s|d|f|n|r... |
```

```

flow-type ether|ip4|tcp4|udp4|sctp4|ah4|esp4|ip6|tcp6|udp6|ah6|esp6|sctp6
[src xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]] [dst xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]]
[proto N [m N]] [src-ip ip-address [m ip-address]] [dst-ip ip-address [m ip-address]]
[tos N [m N]] [tclass N [m N]] [l4proto N [m N]] [src-port N [m N]] [dst-port N [m N]]
[spi N [m N]] [l4data N [m N]] [vlan-etype N [m N]] [vlan N [m N]] [user-def N [m N]] [dst-
mac xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]] [action N] [context N] [loc N] |
delete N

ethtool -w|--get-dump devname [data filename]
ethtool -W|--set-dump devname N
ethtool -T|--show-time-stamping devname
ethtool -x|--show-rxfh-indir|--show-rxfh devname
ethtool -X|--set-rxfh-indir|--rxfh devname [hkey xx:yy:zz:aa:bb:cc:...] [start N] [ equal N |
    weight W0 W1 ... | default ] [hfunc FUNC] [context CTX | new] [delete]
ethtool -f|--flash devname file [N]
ethtool -l|--show-channels devname
ethtool -L|--set-channels devname [rx N] [tx N] [other N] [combined N]
ethtool -m|--dump-module-eeprom|--module-info devname [raw on|off] [hex on|off] [offset N]
    [length N] [page N] [bank N] [i2c N]
ethtool --show-priv-flags devname
ethtool --set-priv-flags devname flag on|off ...
ethtool --show-eee devname
ethtool --set-eee devname [eee on|off] [tx-lpi on|off] [tx-timer N] [advertise N]
ethtool --set-phy-tunable devname [ downshift on|off [count N] ] [ fast-link-down on|off [msecs N] ]
    [ energy-detect-power-down on|off [msecs N] ]
ethtool --get-phy-tunable devname [downshift] [fast-link-down] [energy-detect-power-down]
ethtool --get-tunable devname [rx-copybreak] [tx-copybreak] [pfc-prevention-tout]
ethtool --set-tunable devname [rx-copybreak N] [tx-copybreak N] [pfc-prevention-tout N]
ethtool --reset devname [flags N] [mgmt] [mgmt-shared] [irq] [irq-shared] [dma] [dma-shared]
    [filter] [filter-shared] [offload] [offload-shared] [mac] [mac-shared] [phy] [phy-shared] [ram]
    [ram-shared] [ap] [ap-shared] [dedicated] [all]
ethtool --show-fec devname
ethtool --set-fec devname encoding auto|off|rs|baser|llrs [...]
ethtool -Q|--per-queue devname [queue_mask %ox] sub_command ...

ethtool --cable-test devname
ethtool --cable-test-tdr devname [first N] [last N] [step N] [pair N]
ethtool --show-tunnels devname

```

DESCRIPTION

ethtool is used to query and control network device driver and hardware settings, particularly for wired Ethernet devices.

devname is the name of the network device on which ethtool should operate.

OPTIONS

ethtool with a single argument specifying the device name prints current settings of the specified device.

-h --help

Shows a short help message.

--version

Shows the ethtool version number.

--debug *N*

Turns on debugging messages. Argument is interpreted as a mask:

0x01 Parser information

--json Output results in JavaScript Object Notation (JSON). Only a subset of options support this. Those which do not will continue to output plain text in the presence of this option.

-I --include-statistics

Include command-related statistics in the output. This option allows displaying relevant device statistics for selected get commands.

-a --show-pause

Queries the specified Ethernet device for pause parameter information.

-A --pause

Changes the pause parameters of the specified Ethernet device.

autoneg on|off

Specifies whether pause autonegotiation should be enabled.

rx on|off

Specifies whether RX pause should be enabled.

tx on|off

Specifies whether TX pause should be enabled.

-c --show-coalesce

Queries the specified network device for coalescing information.

-C --coalesce

Changes the coalescing settings of the specified network device.

-g --show-ring

Queries the specified network device for rx/tx ring parameter information.

-G --set-ring

Changes the rx/tx ring parameters of the specified network device.

rx *N* Changes the number of ring entries for the Rx ring.

rx-mini *N*

Changes the number of ring entries for the Rx Mini ring.

rx-jumbo *N*

Changes the number of ring entries for the Rx Jumbo ring.

tx *N* Changes the number of ring entries for the Tx ring.

-i --driver

Queries the specified network device for associated driver information.

-d --register-dump

Retrieves and prints a register dump for the specified network device. The register format for some devices is known and decoded others are printed in hex. When *aw* is enabled, then ethtool dumps the raw register data to stdout. If *file* is specified, then use contents of previous raw register dump, rather than reading from the device.

-e --eeprom-dump

Retrieves and prints an EEPROM dump for the specified network device. When raw is enabled, then it dumps the raw EEPROM data to stdout. The length and offset parameters allow dumping certain portions of the EEPROM. Default is to dump the entire EEPROM.

raw on/off**offset N****length N****-E --change-eeprom**

If value is specified, changes EEPROM byte for the specified network device. offset and value specify which byte and its new value. If value is not specified, stdin is read and written to the EEPROM. The length and offset parameters allow writing to certain portions of the EEPROM. Because of the persistent nature of writing to the EEPROM, a device-specific magic key must be specified to prevent the accidental writing to the EEPROM.

-k --show-features --show-offload

Queries the specified network device for the state of protocol offload and other features.

-K --features --offload

Changes the offload parameters and other features of the specified network device. The following feature names are built-in and others may be defined by the kernel.

rx on|off

Specifies whether RX checksumming should be enabled.

tx on|off

Specifies whether TX checksumming should be enabled.

sg on|off

Specifies whether scatter-gather should be enabled.

tso on|off

Specifies whether TCP segmentation offload should be enabled.

ufo on|off

Specifies whether UDP fragmentation offload should be enabled

gso on|off

Specifies whether generic segmentation offload should be enabled

gro on|off

Specifies whether generic receive offload should be enabled

lro on|off

Specifies whether large receive offload should be enabled

rxvlan on|off

Specifies whether RX VLAN acceleration should be enabled

txvlan on|off

Specifies whether TX VLAN acceleration should be enabled

ntuple on|off

Specifies whether Rx ntuple filters and actions should be enabled

rxhash on|off

Specifies whether receive hashing offload should be enabled

-p --identify

Initiates adapter-specific action intended to enable an operator to easily identify the adapter by sight. Typically this involves blinking one or more LEDs on the specific network port.

[*N*] Length of time to perform phys-id, in seconds.

-P --show-permaddr

Queries the specified network device for permanent hardware address.

-r --negotiate

Restarts auto-negotiation on the specified Ethernet device, if auto-negotiation is enabled.

-S --statistics

Queries the specified network device for standard (IEEE, IETF, etc.), or NIC- and driver-specific statistics. NIC- and driver-specific statistics are requested when no group of statistics is specified.

NIC- and driver-specific statistics and standard statistics are independent, devices may implement either, both or none. There is little commonality between naming of NIC- and driver-specific statistics across vendors.

--all-groups

--groups [eth-phy] [eth-mac] [eth-ctrl] [rmon]

Request groups of standard device statistics.

--phy-statistics

Queries the specified network device for PHY specific statistics.

-t --test

Executes adapter selftest on the specified network device. Possible test modes are:

offline Perform full set of tests, possibly interrupting normal operation during the tests,

online Perform limited set of tests, not interrupting normal operation,

external_lb

Perform full set of tests, as for **offline**, and additionally an external-loopback test.

-s --change

Allows changing some or all settings of the specified network device. All following options only apply if **-s** was specified.

speed *N*

Set speed in Mb/s. **ethtool** with just the device name as an argument will show you the supported device speeds.

lanes *N* Set number of lanes.

duplex half|full

Sets full or half duplex mode.

port tp|auj|bnc|mii

Selects device port.

master-slave preferred-master|preferred-slave|forced-master|forced-slave

Configure MASTER/SLAVE role of the PHY. When the PHY is configured as MASTER, the PMA Transmit function shall source TX_TCLK from a local clock source. When configured as SLAVE, the PMA Transmit function shall source TX_TCLK from the clock recovered from data stream provided by MASTER. Not all devices support this.

preferred-master Prefer MASTER role on autonegotiation

preferred-slave Prefer SLAVE role on autonegotiation

forced-master Force the PHY in MASTER role. Can be used without autonegotiation

forced-slave Force the PHY in SLAVE role. Can be used without autonegotiation

mdix auto|on|off

Selects MDI-X mode for port. May be used to override the automatic detection feature of most adapters. An argument of **auto** means automatic detection of MDI status, **on** forces MDI-X (crossover) mode, while **off** means MDI (straight through) mode. The driver should

guarantee that this command takes effect immediately, and if necessary may reset the link to cause the change to take effect.

autoneg on|off

Specifies whether autonegotiation should be enabled. Autonegotiation is enabled by default, but in some network devices may have trouble with it, so you can disable it if really necessary.

advertise *N*

Sets the speed and duplex advertised by autonegotiation. The argument is a hexadecimal value using one or a combination of the following values:

0x001	10baseT Half	
0x002	10baseT Full	
0x004	100baseT Half	
0x008	100baseT Full	
0x8000000000000000	100baseT1 Full	
0x4000000000000000	100baseFX Half	
0x8000000000000000	100baseFX Full	
0x010	1000baseT Half	(not supported by IEEE standards)
0x020	1000baseT Full	
0x20000	1000baseKX Full	
0x200000000000	1000baseX Full	
0x1000000000000000	1000baseT1 Full	
0x8000	2500baseX Full	(not supported by IEEE standards)
0x80000000000000	2500baseT Full	
0x10000000000000	5000baseT Full	
0x1000	10000baseT Full	
0x40000	10000baseKX4 Full	
0x80000	10000baseKR Full	
0x100000	10000baseR_FEC	
0x400000000000	10000baseCR Full	
0x800000000000	10000baseSR Full	
0x10000000000000	10000baseLR Full	
0x20000000000000	10000baseLRM Full	
0x40000000000000	10000baseER Full	
0x200000	20000baseMLD2 Full	(not supported by IEEE standards)
0x400000	20000baseKR2 Full	(not supported by IEEE standards)
0x80000000	25000baseCR Full	
0x100000000	25000baseKR Full	
0x200000000	25000baseSR Full	
0x80000000	40000baseKR4 Full	
0x10000000	40000baseCR4 Full	
0x20000000	40000baseSR4 Full	
0x40000000	40000baseLR4 Full	
0x4000000000	50000baseCR2 Full	
0x8000000000	50000baseKR2 Full	
0x100000000000	50000baseSR2 Full	
0x10000000000000	50000baseKR Full	
0x20000000000000	50000baseSR Full	
0x40000000000000	50000baseCR Full	
0x80000000000000	50000baseLR_ER_FR Full	
0x1000000000000000	50000baseDR Full	
0x8000000	56000baseKR4 Full	
0x100000000	56000baseCR4 Full	
0x200000000	56000baseSR4 Full	

0x40000000	56000baseLR4 Full
0x1000000000	100000baseKR4 Full
0x2000000000	100000baseSR4 Full
0x4000000000	100000baseCR4 Full
0x8000000000	100000baseLR4_ER4 Full
0x2000000000000000	100000baseKR2 Full
0x4000000000000000	100000baseSR2 Full
0x8000000000000000	100000baseCR2 Full
0x1000000000000000	100000baseLR2_ER2_FR2 Full
0x2000000000000000	100000baseDR2 Full
0x8000000000000000	100000baseKR Full
0x1000000000000000	100000baseSR Full
0x2000000000000000	100000baseLR_ER_FR Full
0x4000000000000000	100000baseCR Full
0x8000000000000000	100000baseDR Full
0x4000000000000000	200000baseKR4 Full
0x8000000000000000	200000baseSR4 Full
0x1000000000000000	200000baseLR4_ER4_FR4 Full
0x2000000000000000	200000baseDR4 Full
0x4000000000000000	200000baseCR4 Full
0x1000000000000000	200000baseKR2 Full
0x2000000000000000	200000baseSR2 Full
0x4000000000000000	200000baseLR2_ER2_FR2 Full
0x8000000000000000	200000baseDR2 Full
0x1000000000000000	200000baseCR2 Full
0x2000000000000000	400000baseKR8 Full
0x4000000000000000	400000baseSR8 Full
0x8000000000000000	400000baseLR8_ER8_FR8 Full
0x1000000000000000	400000baseDR8 Full
0x2000000000000000	400000baseCR8 Full
0x4000000000000000	400000baseKR4 Full
0x8000000000000000	400000baseSR4 Full
0x1000000000000000	400000baseLR4_ER4_FR4 Full
0x2000000000000000	400000baseDR4 Full
0x4000000000000000	400000baseCR4 Full

phyad *N*

PHY address.

xcvr internal|external

Selects transceiver type. Currently only internal and external can be specified, in the future further types might be added.

wol p|u|m|b|a|g|s|f|d...

Sets Wake-on-LAN options. Not all devices support this. The argument to this option is a string of characters specifying which options to enable.

- p** Wake on PHY activity
- u** Wake on unicast messages
- m** Wake on multicast messages
- b** Wake on broadcast messages
- a** Wake on ARP
- g** Wake on MagicPacket™
- s** Enable SecureOn™ password for MagicPacket™
- f** Wake on filter(s)
- d** Disable (wake on nothing). This option clears all previous options.

sopass xx:yy:zz:aa:bb:cc

Sets the SecureOn™ password. The argument to this option must be 6 bytes in Ethernet MAC hex format (xx:yy:zz:aa:bb:cc).

msglvl *N***msglvl *type on|off ...***

Sets the driver message type flags by name or number. *type* names the type of message to enable or disable; *N* specifies the new flags numerically. The defined type names and numbers are:

drv	0x0001	General driver status
probe	0x0002	Hardware probing
link	0x0004	Link state
timer	0x0008	Periodic status check
ifdown	0x0010	Interface being brought down
ifup	0x0020	Interface being brought up
rx_err	0x0040	Receive error
tx_err	0x0080	Transmit error
tx_queued	0x0100	Transmit queueing
intr	0x0200	Interrupt handling
tx_done	0x0400	Transmit completion
rx_status	0x0800	Receive completion
pktdata	0x1000	Packet contents
hw	0x2000	Hardware status
wol	0x4000	Wake-on-LAN status

The precise meanings of these type flags differ between drivers.

-n -u --show-nfc --show-ntuple

Retrieves receive network flow classification options or rules.

rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6

Retrieves the hash options for the specified flow type.

tcp4	TCP over IPv4
udp4	UDP over IPv4
ah4	IPSEC AH over IPv4
esp4	IPSEC ESP over IPv4
sctp4	SCTP over IPv4
tcp6	TCP over IPv6
udp6	UDP over IPv6
ah6	IPSEC AH over IPv6
esp6	IPSEC ESP over IPv6
sctp6	SCTP over IPv6

rule *N*

Retrieves the RX classification rule with the given ID.

-N -U --config-nfc --config-ntuple

Configures receive network flow classification options or rules.

rx-flow-hash tcp4|udp4|ah4|esp4|sctp4|tcp6|udp6|ah6|esp6|sctp6m|v|t|s|d|f|n|r...

Configures the hash options for the specified flow type.

- m** Hash on the Layer 2 destination address of the rx packet.
- v** Hash on the VLAN tag of the rx packet.
- t** Hash on the Layer 3 protocol field of the rx packet.
- s** Hash on the IP source address of the rx packet.
- d** Hash on the IP destination address of the rx packet.
- f** Hash on bytes 0 and 1 of the Layer 4 header of the rx packet.
- n** Hash on bytes 2 and 3 of the Layer 4 header of the rx packet.

r Discard all packets of this flow type. When this option is set, all other options are ignored.

flow-type ether|ip4|tcp4|udp4|sctp4|ah4|esp4|ip6|tcp6|udp6|ah6|esp6|sctp6

Inserts or updates a classification rule for the specified flow type.

ether	Ethernet
ip4	Raw IPv4
tcp4	TCP over IPv4
udp4	UDP over IPv4
sctp4	SCTP over IPv4
ah4	IPSEC AH over IPv4
esp4	IPSEC ESP over IPv4
ip6	Raw IPv6
tcp6	TCP over IPv6
udp6	UDP over IPv6
sctp6	SCTP over IPv6
ah6	IPSEC AH over IPv6
esp6	IPSEC ESP over IPv6

For all fields that allow both a value and a mask to be specified, the mask may be specified immediately after the value using the **m** keyword, or separately using the field name keyword with **-mask** appended, e.g. **src-mask**.

src xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]

Includes the source MAC address, specified as 6 bytes in hexadecimal separated by colons, along with an optional mask. Valid only for flow-type ether.

dst xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]

Includes the destination MAC address, specified as 6 bytes in hexadecimal separated by colons, along with an optional mask. Valid only for flow-type ether.

proto N [m N]

Includes the Ethernet protocol number (ethertype) and an optional mask. Valid only for flow-type ether.

src-ip ip-address [m ip-address]

Specify the source IP address of the incoming packet to match along with an optional mask. Valid for all IP based flow-types.

dst-ip ip-address [m ip-address]

Specify the destination IP address of the incoming packet to match along with an optional mask. Valid for all IP based flow-types.

tos N [m N]

Specify the value of the Type of Service field in the incoming packet to match along with an optional mask. Applies to all IPv4 based flow-types.

tclass N [m N]

Specify the value of the Traffic Class field in the incoming packet to match along with an optional mask. Applies to all IPv6 based flow-types.

l4proto N [m N]

Includes the layer 4 protocol number and optional mask. Valid only for flow-types ip4 and ip6.

src-port N [m N]

Specify the value of the source port field (applicable to TCP/UDP packets) in the incoming packet to match along with an optional mask. Valid for flow-types ip4, tcp4, udp4, and sctp4 and their IPv6 equivalents.

dst-port N [m N]

Specify the value of the destination port field (applicable to TCP/UDP packets) in the incoming packet to match along with an optional mask. Valid for flow-types ip4, tcp4, udp4, and sctp4 and their IPv6 equivalents.

spi N [m N]

Specify the value of the security parameter index field (applicable to AH/ESP packets) in the incoming packet to match along with an optional mask. Valid for flow-types ip4, ah4, and esp4 and their IPv6 equivalents.

l4data N [m N]

Specify the value of the first 4 Bytes of Layer 4 in the incoming packet to match along with an optional mask. Valid for ip4 and ip6 flow-types.

vlan-etype N [m N]

Includes the VLAN tag Ethertype and an optional mask.

vlan N [m N]

Includes the VLAN tag and an optional mask.

user-def N [m N]

Includes 64-bits of user-specific data and an optional mask.

dst-mac xx:yy:zz:aa:bb:cc [m xx:yy:zz:aa:bb:cc]

Includes the destination MAC address, specified as 6 bytes in hexadecimal separated by colons, along with an optional mask. Valid for all IP based flow-types.

action N

Specifies the Rx queue to send packets to, or some other action.

-1 Drop the matched flow

-2 Use the matched flow as a Wake-on-LAN filter

0 or higher Rx queue to route the flow

context N

Specifies the RSS context to spread packets over multiple queues; either **0** for the default RSS context, or a value returned by **ethtool -X ... context new**.

vf N

Specifies the Virtual Function the filter applies to. Not compatible with action.

queue N

Specifies the Rx queue to send packets to. Not compatible with action.

loc N

Specify the location/ID to insert the rule. This will overwrite any rule present in that location and will not go through any of the rule ordering process.

delete N

Deletes the RX classification rule with the given ID.

-w --get-dump

Retrieves and prints firmware dump for the specified network device. By default, it prints out the dump flag, version and length of the dump data. When *data* is indicated, then ethtool fetches the dump data and directs it to a *file*.

-W --set-dump

Sets the dump flag for the device.

-T --show-time-stamping

Show the device's time stamping capabilities and associated PTP hardware clock.

-x --show-rxfh-indir --show-rxfh

Retrieves the receive flow hash indirection table and/or RSS hash key.

-X --set-rxh-indir --rxh

Configures the receive flow hash indirection table and/or RSS hash key.

hkey Sets RSS hash key of the specified network device. RSS hash key should be of device supported length. Hash key format must be in xx:yy:zz:aa:bb:cc format meaning both the nibbles of a byte should be mentioned even if a nibble is zero.

hfunc Sets RSS hash function of the specified network device. List of RSS hash functions which kernel supports is shown as a part of the --show-rxh command output.

start *N* For the **equal** and **weight** options, sets the starting receive queue for spreading flows to *N*.

equal *N*

Sets the receive flow hash indirection table to spread flows evenly between the first *N* receive queues.

weight *W0 W1 ...*

Sets the receive flow hash indirection table to spread flows between receive queues according to the given weights. The sum of the weights must be non-zero and must not exceed the size of the indirection table.

default Sets the receive flow hash indirection table to its default value.

context *CTX* | **new**

Specifies an RSS context to act on; either **new** to allocate a new RSS context, or *CTX*, a value returned by a previous ... **context new**.

delete Delete the specified RSS context. May only be used in conjunction with**context** and a non-zero *CTX* value.

-f --flash

Write a firmware image to flash or other non-volatile memory on the device.

file Specifies the filename of the firmware image. The firmware must first be installed in one of the directories where the kernel firmware loader or firmware agent will look, such as /lib/firmware.

N If the device stores multiple firmware images in separate regions of non-volatile memory, this parameter may be used to specify which region is to be written. The default is 0, requesting that all regions are written. All other values are driver-dependent.

-l --show-channels

Queries the specified network device for the numbers of channels it has. A channel is an IRQ and the set of queues that can trigger that IRQ.

-L --set-channels

Changes the numbers of channels of the specified network device.

rx *N* Changes the number of channels with only receive queues.

tx *N* Changes the number of channels with only transmit queues.

other *N*

Changes the number of channels used only for other purposes e.g. link interrupts or SR-IOV co-ordination.

combined *N*

Changes the number of multi-purpose channels.

-m --dump-module-eeprom --module-info

Retrieves and if possible decodes the EEPROM from plugin modules, e.g SFP+, QSFP. If the driver and module support it, the optical diagnostic information is also read and decoded. When either one of *page*, *bank* or *i2c* parameters is specified, dumps only of a single page or its portion is allowed. In such a case *offset* and *length* parameters are treated relatively to EEPROM page boundaries.

--show-priv-flags

Queries the specified network device for its private flags. The names and meanings of private flags (if any) are defined by each network device driver.

--set-priv-flags

Sets the device's private flags as specified.

flag **on|off** Sets the state of the named private flag.

--show-eee

Queries the specified network device for its support of Energy-Efficient Ethernet (according to the IEEE 802.3az specifications)

--set-eee

Sets the device EEE behaviour.

eee on|off

Enables/disables the device support of EEE.

tx-lpi on|off

Determines whether the device should assert its Tx LPI.

advertise *N*

Sets the speeds for which the device should advertise EEE capabilities. Values are as for

--change advertise**tx-timer** *N*

Sets the amount of time the device should stay in idle mode prior to asserting its Tx LPI (in microseconds). This has meaning only when Tx LPI is enabled.

--set-phy-tunable

Sets the PHY tunable parameters.

downshift on|off

Specifies whether downshift should be enabled.

count *N*

Sets the PHY downshift re-tries count.

fast-link-down **on|off**

Specifies whether Fast Link Down should be enabled and time until link down (if supported).

msecs *N*

Sets the period after which the link is reported as down. Note that the PHY may choose the closest supported value. Only on reading back the tunable do you get the actual value.

energy-detect-power-down **on|off**

Specifies whether Energy Detect Power Down (EDPD) should be enabled (if supported). This will put the RX and TX circuit blocks into a low power mode, and the PHY will wake up periodically to send link pulses to avoid any lock-up situation with a peer PHY that may also have EDPD enabled. By default, this setting will also enable the periodic transmission of TX pulses.

msecs *N*

Some PHYs support configuration of the wake-up interval to send TX pulses.

This setting allows the control of this interval, and 0 disables TX pulses if the PHY supports this. Disabling TX pulses can create a lock-up situation where neither of the PHYs wakes the other one. If unspecified the default value (in milliseconds) will be used by the PHY.

--get-phy-tunable

Gets the PHY tunable parameters.

downshift

For operation in cabling environments that are incompatible with 1000BASE-T, PHY device provides an automatic link speed downshift operation. Link speed downshift after N failed 1000BASE-T auto-negotiation attempts. Downshift is useful where cable does not have the 4 pairs instance.

Gets the PHY downshift count/status.

fast-link-down

Depending on the mode it may take 0.5s - 1s until a broken link is reported as down. In certain use cases a link-down event needs to be reported as soon as possible. Some PHYs support a Fast Link Down Feature and may allow configuration of the delay before a broken link is reported as being down.

Gets the PHY Fast Link Down status / period.

energy-detect-power-down

Gets the current configured setting for Energy Detect Power Down (if supported).

--get-tunable

Get the tunable parameters.

rx-copybreak

Get the current rx copybreak value in bytes.

tx-copybreak

Get the current tx copybreak value in bytes.

pfc-prevention-tout

Get the current pfc prevention timeout value in msecs.

--set-tunable

Set driver's tunable parameters.

rx-copybreak *N*

Set the rx copybreak value in bytes.

tx-copybreak *N*

Set the tx copybreak value in bytes.

pfc-prevention-tout *N*

Set pfc prevention timeout in msecs. Value of 0 means disable and 65535 means auto.

--reset

Reset hardware components specified by flags and components listed below

flags *N* Resets the components based on direct flags mask

mgmt Management processor

irq Interrupt requester

dma DMA engine

filter Filtering/flow direction

offload Protocol offload

mac Media access controller

phy Transceiver/PHY

ram RAM shared between multiple components **ap** Application Processor

dedicated

All components dedicated to this interface

all

All components used by this interface, even if shared

--show-fec

Queries the specified network device for its support of Forward Error Correction.

--set-fec

Configures Forward Error Correction for the specified network device.

Forward Error Correction modes selected by a user are expected to be persisted after any hotplug events. If a module is swapped that does not support the current FEC mode, the driver or firmware must take the link down administratively and report the problem in the system logs for users to correct.

encoding auto|off|rs|baser|llrs [...]

Sets the FEC encoding for the device. Combinations of options are specified as e.g. **encoding auto rs**; the semantics of such combinations vary between drivers.

auto Use the driver's default encoding

off Turn off FEC

RS Force RS-FEC encoding

BaseR Force BaseR encoding

LLRS Force LLRS-FEC encoding

-Q|--per-queue

Applies provided sub command to specific queues.

queue_mask %x

Sets the specific queues which the sub command is applied to. If queue_mask is not set, the sub command will be applied to all queues.

sub_command

Sub command to apply. The supported sub commands include --show-coalesce and --coalesce.

q.B --cable-test

Perform a cable test and report the results. What results are returned depends on the capabilities of the network interface. Typically open pairs and shorted pairs can be reported, along with pairs being O.K. When a fault is detected the approximate distance to the fault may be reported.

--cable-test-tdr

Perform a cable test and report the raw Time Domain Reflectometer data. A pulse is sent down a cable pair and the amplitude of the reflection, for a given distance, is reported. A break in the cable returns a big reflection. Minor damage to the cable returns a small reflection. If the cable is shorted, the amplitude of the reflection can be negative. By default, data is returned for lengths between 0 and 150m at 1m steps, for all pairs. However parameters can be passed to restrict the collection of data. It should be noted, that the interface will round the distances to whatever granularity is actually implemented. This is often 0.8 of a meter. The results should include the actual rounded first and last distance and step size.

first N Distance along the cable, in meters, where the first measurement should be made.

last N Distance along the cable, in meters, where the last measurement should be made.

step N Distance, in meters, between each measurement.

pair N Which pair should be measured. Typically a cable has 4 pairs. 0 = Pair A, 1 = Pair B, ...

--monitor

Listens to netlink notification and displays them.

command

If argument matching a command is used, ethtool only shows notifications of this type. Without such argument or with --all, all notification types are shown.

devname

If a device name is used as argument, only notification for this device are shown. Default is to show notifications for all devices.

--show-tunnels

Show tunnel-related device capabilities and state. List UDP ports kernel has programmed the device to parse as VxLAN, or GENEVE tunnels.

BUGS

Not supported (in part or whole) on all network drivers.

AUTHOR

ethtool was written by David Miller.

Modifications by Jeff Garzik, Tim Hockin, Jakub Jelinek, Andre Majorel, Eli Kupermann, Scott Feldman, Andi Kleen, Alexander Duyck, Sucheta Chakraborty, Jesse Brandenburg, Ben Hutchings, Scott Branden.

AVAILABILITY

ethtool is available from <<http://www.kernel.org/pub/software/network/ethtool/>>

NAME

exfatlabel – Get or Set volume label or volume serial of an exFAT filesystem

SYNOPSIS

exfatlabel [**-i** *volume-label*] [**-v**] *device* [*label_string or serial_value*]
exfatlabel -V

DESCRIPTION

exfatlabel Print or set volume label of an existing exFAT filesystem.

If there is a *label_string* in argument of exfatlabel, It will be written to volume label field on given device. If not, exfatlabel will just print out after reading volume label field from given device. If -i or --volume-serial is given, It can be switched to volume serial mode.

OPTIONS

- i** Switch to volume serial mode.
- V** Prints the version number and exits.

exports(5) - Linux man page

Name

exports - NFS server export table

Description

The file */etc/exports* contains a table of local physical file systems on an NFS server that are accessible to NFS clients. The contents of the file are maintained by the server's system administrator.

Each file system in this table has a list of options and an access control list. The table is used by **exportfs(8)** to give information to **mountd(8)**.

The file format is similar to the SunOS *exports* file. Each line contains an export point and a whitespace-separated list of clients allowed to mount the file system at that point. Each listed client may be immediately followed by a parenthesized, comma-separated list of export options for that client. No whitespace is permitted between a client and its option list.

Also, each line may have one or more specifications for default options after the path name, in the form of a dash ("") followed by an option list. The option list is used for all subsequent exports on that line only.

Blank lines are ignored. A pound sign ("#") introduces a comment to the end of the line. Entries may be continued across newlines using a backslash. If an export name contains spaces it should be quoted using double quotes. You can also specify spaces or other unusual character in the export name using a backslash followed by the character code as three octal digits.

To apply changes to this file, run **exportfs**-ra or restart the NFS server.

Machine Name Formats

NFS clients may be specified in a number of ways:

single host

You may specify a host either by an abbreviated name recognized by the resolver, the fully qualified domain name, an IPv4 address, or an IPv6 address. IPv6 addresses must not be inside square brackets in */etc/exports* lest they be confused with character-class wildcard matches.

netgroups

NIS netgroups may be given as *@group*. Only the host part of each netgroup members is considered in checking for membership. Empty host parts or those containing a single dash (-) are ignored.

wildcards

+Machine names may contain the wildcard characters * and ?, or may contain character class lists within [square brackets]. This can be used to make the *exports* file more compact; for instance, *.cs.foo.edu matches all hosts in the domain cs.foo.edu. As these characters also match the dots in a domain name, the given pattern will also match all hosts within any subdomain of cs.foo.edu.

IP networks

You can also export directories to all hosts on an IP (sub-) network simultaneously. This is done by specifying an IP address and netmask pair as *address/netmask* where the netmask can be specified in dotted-decimal format, or as a contiguous mask length. For example, either '/255.255.252.0' or '/22' appended to the network base IPv4 address results in identical subnetworks with 10 bits of host. IPv6 addresses must use a contiguous mask length and must not be inside square brackets to avoid confusion with character-class wildcards. Wildcard characters generally do not work on IP addresses, though they may work by accident when reverse DNS lookups fail.

RPCSEC_GSS security

You may use the special strings "gss/krb5", "gss/krb5i", or "gss/krb5p" to restrict access to clients using rpcsec_gss security. However, this syntax is deprecated; on linux kernels since 2.6.23, you should instead use the "sec=" export option:

sec=

The *sec=* option, followed by a colon-delimited list of security flavors, restricts the export to clients using those flavors. Available security flavors include sys (the default--no cryptographic security), krb5 (authentication only), krb5i (integrity protection), and krb5p (privacy protection). For the purposes of security flavor negotiation, order counts: preferred flavors should be listed first. The order of the *sec=* option with respect to the other options does not matter, unless you want some options to be enforced differently depending on flavor. In that case you may include multiple *sec=* options, and following options will be enforced only for access using flavors listed in the immediately preceding *sec=* option. The only options that are permitted to vary in this way are *ro*, *rw*, *no_root_squash*, *root_squash*, and *all_squash*.

General Options

exports understands the following export options:

secure

This option requires that requests originate on an Internet port less than IPPORT_RESERVED (1024). This option is on by default. To turn it off, specify *insecure*.

rw

Allow both read and write requests on this NFS volume. The default is to disallow any request which changes the filesystem. This can also be made explicit by using the *ro*

option.

async

This option allows the NFS server to violate the NFS protocol and reply to requests before any changes made by that request have been committed to stable storage (e.g. disc drive).

Using this option usually improves performance, but at the cost that an unclean server restart (i.e. a crash) can cause data to be lost or corrupted.

sync

Reply to requests only after the changes have been committed to stable storage (see *async* above).

In releases of nfs-utils up to and including 1.0.0, the *async* option was the default. In all releases after 1.0.0, *sync* is the default, and *async* must be explicitly requested if needed. To help make system administrators aware of this change, **exportfs** will issue a warning if neither *sync* nor *async* is specified.

no_wdelay

This option has no effect if *async* is also set. The NFS server will normally delay committing a write request to disc slightly if it suspects that another related write request may be in progress or may arrive soon. This allows multiple write requests to be committed to disc with the one operation which can improve performance. If an NFS server received mainly small unrelated requests, this behaviour could actually reduce performance, so *no_wdelay* is available to turn it off. The default can be explicitly requested with the *wdelay* option.

nohide

This option is based on the option of the same name provided in IRIX NFS. Normally, if a server exports two filesystems one of which is mounted on the other, then the client will have to mount both filesystems explicitly to get access to them. If it just mounts the parent, it will see an empty directory at the place where the other filesystem is mounted. That filesystem is "hidden".

Setting the *nohide* option on a filesystem causes it not to be hidden, and an appropriately authorised client will be able to move from the parent to that filesystem without noticing the change.

However, some NFS clients do not cope well with this situation as, for instance, it is then possible for two files in the one apparent filesystem to have the same inode number.

The *nohide* option is currently only effective on *single host* exports. It does not work reliably with netgroup, subnet, or wildcard exports.

This option can be very useful in some situations, but it should be used with due care, and only after confirming that the client system copes with the situation effectively.

The option can be explicitly disabled with *hide*.

crossmnt

This option is similar to *nohide* but it makes it possible for clients to move from the filesystem marked with *crossmnt* to exported filesystems mounted on it. Thus when a child filesystem "B" is mounted on a parent "A", setting *crossmnt* on "A" has the same effect as setting "*nohide*" on B.

no_subtree_check

This option disables subtree checking, which has mild security implications, but can improve reliability in some circumstances.

If a subdirectory of a filesystem is exported, but the whole filesystem isn't then whenever a NFS request arrives, the server must check not only that the accessed file is in the appropriate filesystem (which is easy) but also that it is in the exported tree (which is harder). This check is called the *subtree_check*.

In order to perform this check, the server must include some information about the location of the file in the "filehandle" that is given to the client. This can cause problems with accessing files that are renamed while a client has them open (though in many simple cases it will still work).

Subtree checking is also used to make sure that files inside directories to which only root has access can only be accessed if the filesystem is exported with *no_root_squash* (see below), even if the file itself allows more general access.

As a general guide, a home directory filesystem, which is normally exported at the root and may see lots of file renames, should be exported with subtree checking disabled. A filesystem which is mostly readonly, and at least doesn't see many file renames (e.g. /usr or /var) and for which subdirectories may be exported, should probably be exported with subtree checks enabled.

The default of having subtree checks enabled, can be explicitly requested with *subtree_check*.

From release 1.1.0 of nfs-utils onwards, the default will be *no_subtree_check* as subtree_checking tends to cause more problems than it is worth. If you genuinely require subtree checking, you should explicitly put that option in the **exports** file. If you put neither option, **exportfs** will warn you that the change is pending.

insecure_locks

no_auth_nlm

This option (the two names are synonymous) tells the NFS server not to require authentication of locking requests (i.e. requests which use the NLM protocol).

Normally the NFS server will require a lock request to hold a credential for a user who has read access to the file. With this flag no access checks will be performed.

Early NFS client implementations did not send credentials with lock requests, and many current NFS clients still exist which are based on the old implementations. Use this flag if you find that you can only lock files which are world readable.

The default behaviour of requiring authentication for NLM requests can be explicitly requested with either of the synonymous *auth_nlm*, or *secure_locks*.

no_acl

On some specially patched kernels, and when exporting filesystems that support ACLs, this option tells **nfsd** not to reveal ACLs to clients, so they will see only a subset of actual permissions on the given file system. This option is safe for filesystems used by NFSv2 clients and old NFSv3 clients that perform access decisions locally. Current NFSv3 clients use the ACCESS RPC to perform all access decisions on the server. Note that the *no_acl* option only has effect on kernels specially patched to support it, and when exporting filesystems with ACL support. The default is to export with ACL support (i.e. by default, *no_acl* is off).

mountpoint=path *mp*

This option makes it possible to only export a directory if it has successfully been mounted. If no path is given (e.g. *mountpoint* or *mp*) then the export point must also be a mount point. If it isn't then the export point is not exported. This allows you to be sure that the directory underneath a mountpoint will never be exported by accident if, for example, the filesystem failed to mount due to a disc error.

If a path is given (e.g. *mountpoint=/path* or *mp=/path*) then the nominated path must be a mountpoint for the exportpoint to be exported.

fsid=num|root|uuid

NFS needs to be able to identify each filesystem that it exports. Normally it will use a UUID for the filesystem (if the filesystem has such a thing) or the device number of the device holding the filesystem (if the filesystem is stored on the device).

As not all filesystems are stored on devices, and not all filesystems have UUIDs, it is sometimes necessary to explicitly tell NFS how to identify a filesystem. This is done with the *fsid=* option.

For NFSv4, there is a distinguished filesystem which is the root of all exported filesystem. This is specified with *fsid=root* or *fsid=0* both of which mean exactly the same thing.

Other filesystems can be identified with a small integer, or a UUID which should contain 32 hex digits and arbitrary punctuation.

Linux kernels version 2.6.20 and earlier do not understand the UUID setting so a small integer must be used if an fsid option needs to be set for such kernels. Setting

both a small number and a UUID is supported so the same configuration can be made to work on old and new kernels alike.

refer=path@host[+host][:path@host[+host]]

A client referencing the export point will be directed to choose from the given list an alternative location for the filesystem. (Note that the server must have a mountpoint here, though a different filesystem is not required; so, for example, *mount --bind /path /path* is sufficient.)

replicas=path@host[+host][:path@host[+host]]

If the client asks for alternative locations for the export point, it will be given this list of alternatives. (Note that actual replication of the filesystem must be handled elsewhere.)

User ID Mapping

nfsd bases its access control to files on the server machine on the uid and gid provided in each NFS RPC request. The normal behavior a user would expect is that she can access her files on the server just as she would on a normal file system. This requires that the same uids and gids are used on the client and the server machine. This is not always true, nor is it always desirable.

Very often, it is not desirable that the root user on a client machine is also treated as root when accessing files on the NFS server. To this end, uid 0 is normally mapped to a different id: the so-called anonymous or *nobody* uid. This mode of operation (called 'root squashing') is the default, and can be turned off with *no_root_squash*.

By default, **exportfs** chooses a uid and gid of 65534 for squashed access. These values can also be overridden by the *anonuid* and *anongid* options. Finally, you can map all user requests to the anonymous uid by specifying the *all_squash* option.

Here's the complete list of mapping options:

root_squash

Map requests from uid/gid 0 to the anonymous uid/gid. Note that this does not apply to any other uids or gids that might be equally sensitive, such as user *bin* or group *staff*.

no_root_squash

Turn off root squashing. This option is mainly useful for diskless clients.

all_squash

Map all uids and gids to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, etc. The opposite option is *no_all_squash*, which is the default setting.

anonuid and **anongid**

These options explicitly set the uid and gid of the anonymous account. This option is primarily useful for PC/NFS clients, where you might want all requests appear to be from one user. As an example, consider the export entry for **/home/joe** in the

example section below, which maps all requests to uid 150 (which is supposedly that of user joe).

Example

```
# sample /etc/exports file
/
    master(rw) trusty(rw,no_root_squash)
/projects      proj*.local.domain(rw)
/usr          *.local.domain(ro) @trusted(rw)
/home/joe     pc001(rw,all_squash,anonuid=150,anongid=100)
/pub           *(ro,insecure,all_squash)
/srv/www      -sync,rw server @trusted @external(ro)
/foo           2001:db8:9:e54::/64(rw) 192.0.2.0/24(rw)
/build         buildhost[0-9].local.domain(rw)
```

The first line exports the entire filesystem to machines master and trusty. In addition to write access, all uid squashing is turned off for host trusty. The second and third entry show examples for wildcard hostnames and netgroups (this is the entry '@trusted'). The fourth line shows the entry for the PC/NFS client discussed above. Line 5 exports the public FTP directory to every host in the world, executing all requests under the nobody account. The *insecure* option in this entry also allows clients with NFS implementations that don't use a reserved port for NFS. The sixth line exports a directory read-write to the machine 'server' as well as the '@trusted' netgroup, and read-only to netgroup '@external', all three mounts with the 'sync' option enabled. The seventh line exports a directory to both an IPv6 and an IPv4 subnet. The eighth line demonstrates a character class wildcard match.

Files

/etc/exports

See Also

[exportfs\(8\)](#), [netgroup\(5\)](#), [mountd\(8\)](#), [nfsd\(8\)](#), [showmount\(8\)](#).

Referenced By

[nfs\(5\)](#), [nfsd\(7\)](#)

NAME

ext2 – the second extended file system
ext3 – the third extended file system
ext4 – the fourth extended file system

DESCRIPTION

The second, third, and fourth extended file systems, or ext2, ext3, and ext4 as they are commonly known, are Linux file systems that have historically been the default file system for many Linux distributions. They are general purpose file systems that have been designed for extensibility and backwards compatibility. In particular, file systems previously intended for use with the ext2 and ext3 file systems can be mounted using the ext4 file system driver, and indeed in many modern Linux distributions, the ext4 file system driver has been configured to handle mount requests for ext2 and ext3 file systems.

FILE SYSTEM FEATURES

A file system formatted for ext2, ext3, or ext4 can have some collection of the following file system feature flags enabled. Some of these features are not supported by all implementations of the ext2, ext3, and ext4 file system drivers, depending on Linux kernel version in use. On other operating systems, such as the GNU/HURD or FreeBSD, only a very restrictive set of file system features may be supported in their implementations of ext2.

64bit

Enables the file system to be larger than 2^{32} blocks. This feature is set automatically, as needed, but it can be useful to specify this feature explicitly if the file system might need to be resized larger than 2^{32} blocks, even if it was smaller than that threshold when it was originally created. Note that some older kernels and older versions of e2fsprogs will not support file systems with this ext4 feature enabled.

bigalloc

This ext4 feature enables clustered block allocation, so that the unit of allocation is a power of two number of blocks. That is, each bit in the what had traditionally been known as the block allocation bitmap now indicates whether a cluster is in use or not, where a cluster is by default composed of 16 blocks. This feature can decrease the time spent on doing block allocation and brings smaller fragmentation, especially for large files. The size can be specified using the **mke2fs -C** option.

Warning: The bigalloc feature is still under development, and may not be fully supported with your kernel or may have various bugs. Please see the web page <http://ext4.wiki.kernel.org/index.php/Bigalloc> for details. May clash with delayed allocation (see **nodelalloc** mount option).

This feature requires that the **extent** feature be enabled.

casifold

This ext4 feature provides file system level character encoding support for directories with the casifold (+F) flag enabled. This feature is name-preserving on the disk, but it allows applications to lookup for a file in the file system using an encoding equivalent version of the file name.

dir_index

Use hashed b-trees to speed up name lookups in large directories. This feature is supported by ext3 and ext4 file systems, and is ignored by ext2 file systems.

dir_nlink

Normally, ext4 allows an inode to have no more than 65,000 hard links. This applies to regular files as well as directories, which means that there can be no more than 64,998 subdirectories in a directory (because each of the '.' and '..' entries, as well as the directory entry for the directory in its parent directory counts as a hard link). This feature lifts this limit by causing ext4 to use a link count of 1 to indicate that the number of hard links to a directory is not known when the link count might exceed the maximum count limit.

ea_inode

Normally, a file's extended attributes and associated metadata must fit within the inode or the inode's associated extended attribute block. This feature allows the value of each extended attribute to be placed in the data blocks of a separate inode if necessary, increasing the limit on the size and number of extended attributes per file.

encrypt

Enables support for file-system level encryption of data blocks and file names. The inode metadata (timestamps, file size, user/group ownership, etc.) is *not* encrypted.

This feature is most useful on file systems with multiple users, or where not all files should be encrypted. In many use cases, especially on single-user systems, encryption at the block device layer using dm-crypt may provide much better security.

ext_attr

This feature enables the use of extended attributes. This feature is supported by ext2, ext3, and ext4.

extent

This ext4 feature allows the mapping of logical block numbers for a particular inode to physical blocks on the storage device to be stored using an extent tree, which is a more efficient data structure than the traditional indirect block scheme used by the ext2 and ext3 file systems. The use of the extent tree decreases metadata block overhead, improves file system performance, and decreases the need to run **e2fsck(8)** on the file system. (Note: both **extent** and **extents** are accepted as valid names for this feature for historical/backwards compatibility reasons.)

extra_isize

This ext4 feature reserves a specific amount of space in each inode for extended metadata such as nanosecond timestamps and file creation time, even if the current kernel does not currently need to reserve this much space. Without this feature, the kernel will reserve the amount of space for features it currently needs, and the rest may be consumed by extended attributes.

For this feature to be useful the inode size must be 256 bytes in size or larger.

filetype

This feature enables the storage of file type information in directory entries. This feature is supported by ext2, ext3, and ext4.

flex_bg

This ext4 feature allows the per-block group metadata (allocation bitmaps and inode tables) to be placed anywhere on the storage media. In addition, **mke2fs** will place the per-block group metadata together starting at the first block group of each "flex_bg group". The size of the **flex_bg** group can be specified using the **-G** option.

has_journal

Create a journal to ensure file system consistency even across unclean shutdowns. Setting the file system feature is equivalent to using the **-j** option with **mke2fs** or **tune2fs**. This feature is supported by ext3 and ext4, and ignored by the ext2 file system driver.

huge_file

This ext4 feature allows files to be larger than 2 terabytes in size.

inline_data

Allow data to be stored in the inode and extended attribute area.

journal_dev

This feature is enabled on the superblock found on an external journal device. The block size for the external journal must be the same as the file system which uses it.

The external journal device can be used by a file system by specifying the **-J device=<external-device>** option to **mke2fs(8)** or **tune2fs(8)**.

large_dir

This feature increases the limit on the number of files per directory by raising the maximum size of directories and, for hashed b-tree directories (see **dir_index**), the maximum height of the hashed b-tree used to store the directory entries.

large_file

This feature flag is set automatically by modern kernels when a file larger than 2 gigabytes is created. Very old kernels could not handle large files, so this feature flag was used to prohibit those kernels from mounting file systems that they could not understand.

metadata_csum

This ext4 feature enables metadata checksumming. This feature stores checksums for all of the file system metadata (superblock, group descriptor blocks, inode and block bitmaps, directories, and extent tree blocks). The checksum algorithm used for the metadata blocks is different than the one used for group descriptors with the **uninit_bg** feature. These two features are incompatible and **metadata_csum** will be used preferentially instead of **uninit_bg**.

metadata_csum_seed

This feature allows the file system to store the metadata checksum seed in the superblock, which allows the administrator to change the UUID of a file system using the **metadata_csum** feature while it is mounted.

meta_bg

This ext4 feature allows file systems to be resized on-line without explicitly needing to reserve space for growth in the size of the block group descriptors. This scheme is also used to resize file systems which are larger than 2^{32} blocks. It is not recommended that this feature be set when a file system is created, since this alternate method of storing the block group descriptors will slow down the time needed to mount the file system, and newer kernels can automatically set this feature as necessary when doing an online resize and no more reserved space is available in the resize inode.

mmp

This ext4 feature provides multiple mount protection (MMP). MMP helps to protect the file system from being multiply mounted and is useful in shared storage environments.

project

This ext4 feature provides project quota support. With this feature, the project ID of inode will be managed when the file system is mounted.

quota

Create quota inodes (inode #3 for userquota and inode #4 for group quota) and set them in the superblock. With this feature, the quotas will be enabled automatically when the file system is mounted.

Causes the quota files (i.e., user.quota and group.quota which existed in the older quota design) to be hidden inodes.

resize_inode

This file system feature indicates that space has been reserved so that the block group descriptor table can be extended while resizing a mounted file system. The online resize operation is carried out by the kernel, triggered by **resize2fs(8)**. By default **mke2fs** will attempt to reserve enough space so that the file system may grow to 1024 times its initial size. This can be changed using the **resize** extended option.

This feature requires that the **sparse_super** or **sparse_super2** feature be enabled.

sparse_super

This file system feature is set on all modern ext2, ext3, and ext4 file systems. It indicates that backup copies of the superblock and block group descriptors are present only in a few block groups, not all of them.

sparse_super2

This feature indicates that there will only be at most two backup superblocks and block group descriptors. The block groups used to store the backup superblock(s) and blockgroup descriptor(s) are stored in the superblock, but typically, one will be located at the beginning of block group #1, and one in the last block group in the file system. This feature is essentially a more extreme version of sparse_super and is designed to allow a much larger percentage of the disk to have contiguous blocks available for data files.

stable_inodes

Marks the file system's inode numbers and UUID as stable. **r esize2fs(8)** will not allow shrinking a file system with this feature, nor will **tune2fs(8)** allow changing its UUID. This feature allows the use of specialized encryption settings that make use of the inode numbers and UUID. Note that the **encrypt** feature still needs to be enabled separately. **stable_inodes** is a "compat" feature, so old kernels will allow it.

uninit_bg

This ext4 file system feature indicates that the block group descriptors will be protected using checksums, making it safe for **mke2fs(8)** to create a file system without initializing all of the block groups. The kernel will keep a high watermark of unused inodes, and initialize inode tables and blocks lazily. This feature speeds up the time to check the file system using **e2fsck(8)**, and it also speeds up the time required for **mke2fs(8)** to create the file system.

verity

Enables support for verity protected files. Verity files are readonly, and their data is transparently verified against a Merkle tree hidden past the end of the file. Using the Merkle tree's root hash, a verity file can be efficiently authenticated, independent of the file's size.

This feature is most useful for authenticating important read-only files on read-write file systems. If the file system itself is read-only, then using dm-verity to authenticate the entire block device may provide much better security.

MOUNT OPTIONS

This section describes mount options which are specific to ext2, ext3, and ext4. Other generic mount options may be used as well; see **mount(8)** for details.

Mount options for ext2

The 'ext2' file system is the standard Linux file system. Since Linux 2.5.46, for most mount options the default is determined by the file system superblock. Set them with **tune2fs(8)**.

acl|noacl

Support POSIX Access Control Lists (or not). See the **acl(5)** manual page.

bsddf|minixdf

Set the behavior for the *statfs* system call. The **minixdf** behavior is to return in the *f_blocks* field the total number of blocks of the file system, while the **bsddf** behavior (which is the default) is to subtract the overhead blocks used by the ext2 file system and not available for file storage. Thus

```
% mount /k -o minixdf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2630655	86954	2412169	3%	/k

```
% mount /k -o bsddf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2543714	13	2412169	0%	/k

(Note that this example shows that one can add command line options to the options given in */etc/fstab*.)

check=none or nocheck

No checking is done at mount time. This is the default. This is fast. It is wise to invoke **e2fsck(8)** every now and then, e.g. at boot time. The non-default behavior is unsupported (check=normal and check=strict options have been removed). Note that these mount options don't have to be supported if ext4 kernel driver is used for ext2 and ext3 file systems.

debug Print debugging info upon each (re)mount.

errors={continue|remount-ro|panic}

Define the behavior when an error is encountered. (Either ignore errors and just mark the file system erroneous and continue, or remount the file system read-only, or panic and halt the system.) The default is set in the file system superblock, and can be changed using **tune2fs(8)**.

grpid|bsdgroups and nogrpid|sysvgroups

These options define what group id a newly created file gets. When **grpid** is set, it takes the group id of the directory in which it is created; otherwise (the default) it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

grpquota|noquota|quota|usrquota

The **usrquota** (same as **quota**) mount option enables user quota support on the file system. **grpquota** enables group quotas support. You need the quota utilities to actually enable and manage the quota system.

nouid32

Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values.

oldalloc or orlov

Use old allocator or Orlov allocator for new inodes. Orlov is default.

resgid=n and resuid=n

The ext2 file system reserves a certain percentage of the available space (by default 5%, see **mke2fs(8)** and **tune2fs(8)**). These options determine who can use the reserved blocks. (Roughly: whoever has the specified uid, or belongs to the specified group.)

sb=n Instead of using the normal superblock, use an alternative superblock specified by *n*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system's blocksize, the number of blocks per group, and features such as **sparse_super**.

Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system's layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).

The block number here uses 1 k units. Thus, if you want to use logical block 32768 on a file system with 4 k blocks, use "sb=131072".

user_xattr|nouser_xattr

Support "user." extended attributes (or not).

Mount options for ext3

The ext3 file system is a version of the ext2 file system which has been enhanced with journaling. It supports the same options as ext2 as well as the following additions:

journal_dev=devnum/journal_path=path

When the external journal device's major/minor numbers have changed, these options allow the user to specify the new journal location. The journal device is identified either through its new major/minor numbers encoded in devnum, or via a path to the device.

norecovery/noload

Don't load the journal on mounting. Note that if the file system was not unmounted cleanly, skipping the journal replay will lead to the file system containing inconsistencies that can lead to any number of problems.

data={journal|ordered|writeback}

Specifies the journaling mode for file data. Metadata is always journaled. To use modes other than **ordered** on the root file system, pass the mode to the kernel as boot parameter, e.g. *root-flags=data=journal*.

journal

All data is committed into the journal prior to being written into the main file system.

ordered

This is the default mode. All data is forced directly out to the main file system prior to its metadata being committed to the journal.

writeback

Data ordering is not preserved – data may be written into the main file system after its metadata has been committed to the journal. This is rumoured to be the highest-throughput option. It guarantees internal file system integrity, however it can allow old data to appear in files after a crash and journal recovery.

data_err=ignore

Just print an error message if an error occurs in a file data buffer in ordered mode.

data_err=abort

Abort the journal if an error occurs in a file data buffer in ordered mode.

barrier=0 / barrier=1

This disables / enables the use of write barriers in the jbd code. barrier=0 disables, barrier=1 enables (default). This also requires an IO stack which can support barriers, and if jbd gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

commit=nrsec

Start a journal commit every *nrsec* seconds. The default value is 5 seconds. Zero means default.

user_xattr

Enable Extended User Attributes. See the **attr(5)** manual page.

jqfmt={vfsold|vfsv0|vfsv1}

Apart from the old quota system (as in ext2, jqfmt=vfsold aka version 1 quota) ext3 also supports journaled quotas (version 2 quota). jqfmt=vfsv0 or jqfmt=vfsv1 enables journaled quotas. Journaled quotas have the advantage that even after a crash no quota check is required. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

usrjquota=aquota.user|grpjquota=aquota.group

For journaled quotas (jqfmt=vfsv0 or jqfmt=vfsv1), the mount options **usrjquota=aquota.user** and **grpjquota=aquota.group** are required to tell the quota system which quota database files to use. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

Mount options for ext4

The ext4 file system is an advanced level of the ext3 file system which incorporates scalability and reliability enhancements for supporting large file system.

The options **journal_dev**, **journal_path**, **norecovery**, **noload**, **data**, **commit**, **orlov**, **oldalloc**,

[no]user_xattr, [no]acl, bsddf, minixdf, debug, errors, data_err, grpid, bsdgroups, nogrpid, sysv-groups, resgid, resuid, sb, quota, noquota, nouid32, grpquota, usrquota, usrjquota, grpquota, and jqfmt are backwardly compatible with ext3 or ext2.

journal_checksum | nojournal_checksum

The journal_checksum option enables checksumming of the journal transactions. This will allow the recovery code in e2fsck and the kernel to detect corruption in the kernel. It is a compatible change and will be ignored by older kernels.

journal_async_commit

Commit block can be written to disk without waiting for descriptor blocks. If enabled older kernels cannot mount the device. This will enable 'journal_checksum' internally.

barrier=0 / barrier=1 / barrier / nobarrier

These mount options have the same effect as in ext3. The mount options "barrier" and "nobarrier" are added for consistency with other ext4 mount options.

The ext4 file system enables write barriers by default.

inode_readahead_blks=n

This tuning parameter controls the maximum number of inode table blocks that ext4's inode table readahead algorithm will pre-read into the buffer cache. The value must be a power of 2. The default value is 32 blocks.

stripe=n

Number of file system blocks that mballoc will try to use for allocation size and alignment. For RAID5/6 systems this should be the number of data disks * RAID chunk size in file system blocks.

delalloc

Deferring block allocation until write-out time.

nodealloc

Disable delayed allocation. Blocks are allocated when data is copied from user to page cache.

max_batch_time=usec

Maximum amount of time ext4 should wait for additional file system operations to be batch together with a synchronous write operation. Since a synchronous write operation is going to force a commit and then a wait for the I/O complete, it doesn't cost much, and can be a huge throughput win, we wait for a small amount of time to see if any other transactions can piggyback on the synchronous write. The algorithm used is designed to automatically tune for the speed of the disk, by measuring the amount of time (on average) that it takes to finish committing a transaction. Call this time the "commit time". If the time that the transaction has been running is less than the commit time, ext4 will try sleeping for the commit time to see if other operations will join the transaction. The commit time is capped by the max_batch_time, which defaults to 15000 µs (15 ms). This optimization can be turned off entirely by setting max_batch_time to 0.

min_batch_time=usec

This parameter sets the commit time (as described above) to be at least min_batch_time. It defaults to zero microseconds. Increasing this parameter may improve the throughput of multi-threaded, synchronous workloads on very fast disks, at the cost of increasing latency.

journal_ioprio=prio

The I/O priority (from 0 to 7, where 0 is the highest priority) which should be used for I/O operations submitted by kjournald2 during a commit operation. This defaults to 3, which is a slightly higher priority than the default I/O priority.

abort Simulate the effects of calling ext4_abort() for debugging purposes. This is normally used while remounting a file system which is already mounted.

auto_da_alloc|noauto_da_alloc

Many broken applications don't use fsync() when replacing existing files via patterns such as

```
fd = open("foo.new")/write(fd,...)/close(fd)/ rename("foo.new", "foo")
```

or worse yet

```
fd = open("foo", O_TRUNC)/write(fd,...)/close(fd).
```

If auto_da_alloc is enabled, ext4 will detect the replace-via-rename and replace-via-truncate patterns and force that any delayed allocation blocks are allocated such that at the next journal commit, in the default data=ordered mode, the data blocks of the new file are forced to disk before the rename() operation is committed. This provides roughly the same level of guarantees as ext3, and avoids the "zero-length" problem that can happen when a system crashes before the delayed allocation blocks are forced to disk.

noinit_itable

Do not initialize any uninitialized inode table blocks in the background. This feature may be used by installation CD's so that the install process can complete as quickly as possible; the inode table initialization process would then be deferred until the next time the file system is mounted.

init_itable=n

The lazy itable init code will wait n times the number of milliseconds it took to zero out the previous block group's inode table. This minimizes the impact on system performance while the file system's inode table is being initialized.

discard/nodiscard

Controls whether ext4 should issue discard/TRIM commands to the underlying block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs, but it is off by default until sufficient testing has been done.

block_validity/noblock_validity

This option enables/disables the in-kernel facility for tracking file system metadata blocks within internal data structures. This allows multi-block allocator and other routines to quickly locate extents which might overlap with file system metadata blocks. This option is intended for debugging purposes and since it negatively affects the performance, it is off by default.

dioread_lock/dioread_nolock

Controls whether or not ext4 should use the DIO read locking. If the dioread_nolock option is specified ext4 will allocate uninitialized extent before buffer write and convert the extent to initialized after IO completes. This approach allows ext4 code to avoid using inode mutex, which improves scalability on high speed storages. However this does not work with data journaling and dioread_nolock option will be ignored with kernel warning. Note that dioread_nolock code path is only used for extent-based files. Because of the restrictions this options comprises it is off by default (e.g. dioread_lock).

max_dir_size_kb=n

This limits the size of the directories so that any attempt to expand them beyond the specified limit in kilobytes will cause an ENOSPC error. This is useful in memory-constrained environments, where a very large directory can cause severe performance problems or even provoke the Out Of Memory killer. (For example, if there is only 512 MB memory available, a 176 MB directory may seriously cramp the system's style.)

i_version

Enable 64-bit inode version support. This option is off by default.

nombcache

This option disables use of mbcache for extended attribute deduplication. On systems where extended attributes are rarely or never shared between files, use of mbcache for deduplication adds

unnecessary computational overhead.

prjquota

The prjquota mount option enables project quota support on the file system. You need the quota utilities to actually enable and manage the quota system. This mount option requires the **project** file system feature.

FILE ATTRIBUTES

The ext2, ext3, and ext4 file systems support setting the following file attributes on Linux systems using the **chattr(1)** utility:

a - append only

A - no atime updates

d - no dump

D - synchronous directory updates

i - immutable

S - synchronous updates

u - undeletable

In addition, the ext3 and ext4 file systems support the following flag:

j - data journaling

Finally, the ext4 file system also supports the following flag:

e - extents format

For descriptions of these attribute flags, please refer to the **chattr(1)** man page.

KERNEL SUPPORT

This section lists the file system driver (e.g., ext2, ext3, ext4) and upstream kernel version where a particular file system feature was supported. Note that in some cases the feature was present in earlier kernel versions, but there were known, serious bugs. In other cases the feature may still be considered in an experimental state. Finally, note that some distributions may have backported features into older kernels; in particular the kernel versions in certain "enterprise distributions" can be extremely misleading.

filetype	ext2, 2.2.0
sparse_super	ext2, 2.2.0
large_file	ext2, 2.2.0
has_journal	ext3, 2.4.15
ext_attr	ext2/ext3, 2.6.0
dir_index	ext3, 2.6.0
resize_inode	ext3, 2.6.10 (online resizing)
64bit	ext4, 2.6.28
dir_nlink	ext4, 2.6.28

extent	ext4, 2.6.28
extra_isize	ext4, 2.6.28
flex_bg	ext4, 2.6.28
huge_file	ext4, 2.6.28
meta_bg	ext4, 2.6.28
uninit_bg	ext4, 2.6.28
mmp	ext4, 3.0
bigalloc	ext4, 3.2
quota	ext4, 3.6
inline_data	ext4, 3.8
sparse_super2	ext4, 3.16
metadata_csum	ext4, 3.18
encrypt	ext4, 4.1
metadata_csum_seed	ext4, 4.4
project	ext4, 4.5
ea_inode	ext4, 4.13
large_dir	ext4, 4.13
casefold	ext4, 5.2
verity	ext4, 5.4
stable_inodes	ext4, 5.5

SEE ALSO

mke2fs(8), mke2fs.conf(5), e2fsck(8), dumpe2fs(8), tune2fs(8), debugfs(8), mount(8), chattr(1)

NAME

ext2 – the second extended file system
ext3 – the third extended file system
ext4 – the fourth extended file system

DESCRIPTION

The second, third, and fourth extended file systems, or ext2, ext3, and ext4 as they are commonly known, are Linux file systems that have historically been the default file system for many Linux distributions. They are general purpose file systems that have been designed for extensibility and backwards compatibility. In particular, file systems previously intended for use with the ext2 and ext3 file systems can be mounted using the ext4 file system driver, and indeed in many modern Linux distributions, the ext4 file system driver has been configured to handle mount requests for ext2 and ext3 file systems.

FILE SYSTEM FEATURES

A file system formatted for ext2, ext3, or ext4 can have some collection of the following file system feature flags enabled. Some of these features are not supported by all implementations of the ext2, ext3, and ext4 file system drivers, depending on Linux kernel version in use. On other operating systems, such as the GNU/HURD or FreeBSD, only a very restrictive set of file system features may be supported in their implementations of ext2.

64bit

Enables the file system to be larger than 2^{32} blocks. This feature is set automatically, as needed, but it can be useful to specify this feature explicitly if the file system might need to be resized larger than 2^{32} blocks, even if it was smaller than that threshold when it was originally created. Note that some older kernels and older versions of e2fsprogs will not support file systems with this ext4 feature enabled.

bigalloc

This ext4 feature enables clustered block allocation, so that the unit of allocation is a power of two number of blocks. That is, each bit in the what had traditionally been known as the block allocation bitmap now indicates whether a cluster is in use or not, where a cluster is by default composed of 16 blocks. This feature can decrease the time spent on doing block allocation and brings smaller fragmentation, especially for large files. The size can be specified using the **mke2fs -C** option.

Warning: The bigalloc feature is still under development, and may not be fully supported with your kernel or may have various bugs. Please see the web page <http://ext4.wiki.kernel.org/index.php/Bigalloc> for details. May clash with delayed allocation (see **nodelalloc** mount option).

This feature requires that the **extent** feature be enabled.

casifold

This ext4 feature provides file system level character encoding support for directories with the casifold (+F) flag enabled. This feature is name-preserving on the disk, but it allows applications to lookup for a file in the file system using an encoding equivalent version of the file name.

dir_index

Use hashed b-trees to speed up name lookups in large directories. This feature is supported by ext3 and ext4 file systems, and is ignored by ext2 file systems.

dir_nlink

Normally, ext4 allows an inode to have no more than 65,000 hard links. This applies to regular files as well as directories, which means that there can be no more than 64,998 subdirectories in a directory (because each of the '.' and '..' entries, as well as the directory entry for the directory in its parent directory counts as a hard link). This feature lifts this limit by causing ext4 to use a link count of 1 to indicate that the number of hard links to a directory is not known when the link count might exceed the maximum count limit.

ea_inode

Normally, a file's extended attributes and associated metadata must fit within the inode or the inode's associated extended attribute block. This feature allows the value of each extended attribute to be placed in the data blocks of a separate inode if necessary, increasing the limit on the size and number of extended attributes per file.

encrypt

Enables support for file-system level encryption of data blocks and file names. The inode metadata (timestamps, file size, user/group ownership, etc.) is *not* encrypted.

This feature is most useful on file systems with multiple users, or where not all files should be encrypted. In many use cases, especially on single-user systems, encryption at the block device layer using dm-crypt may provide much better security.

ext_attr

This feature enables the use of extended attributes. This feature is supported by ext2, ext3, and ext4.

extent

This ext4 feature allows the mapping of logical block numbers for a particular inode to physical blocks on the storage device to be stored using an extent tree, which is a more efficient data structure than the traditional indirect block scheme used by the ext2 and ext3 file systems. The use of the extent tree decreases metadata block overhead, improves file system performance, and decreases the need to run **e2fsck(8)** on the file system. (Note: both **extent** and **extents** are accepted as valid names for this feature for historical/backwards compatibility reasons.)

extra_isize

This ext4 feature reserves a specific amount of space in each inode for extended metadata such as nanosecond timestamps and file creation time, even if the current kernel does not currently need to reserve this much space. Without this feature, the kernel will reserve the amount of space for features it currently needs, and the rest may be consumed by extended attributes.

For this feature to be useful the inode size must be 256 bytes in size or larger.

filetype

This feature enables the storage of file type information in directory entries. This feature is supported by ext2, ext3, and ext4.

flex_bg

This ext4 feature allows the per-block group metadata (allocation bitmaps and inode tables) to be placed anywhere on the storage media. In addition, **mke2fs** will place the per-block group metadata together starting at the first block group of each "flex_bg group". The size of the **flex_bg** group can be specified using the **-G** option.

has_journal

Create a journal to ensure file system consistency even across unclean shutdowns. Setting the file system feature is equivalent to using the **-j** option with **mke2fs** or **tune2fs**. This feature is supported by ext3 and ext4, and ignored by the ext2 file system driver.

huge_file

This ext4 feature allows files to be larger than 2 terabytes in size.

inline_data

Allow data to be stored in the inode and extended attribute area.

journal_dev

This feature is enabled on the superblock found on an external journal device. The block size for the external journal must be the same as the file system which uses it.

The external journal device can be used by a file system by specifying the **-J device=<external-device>** option to **mke2fs(8)** or **tune2fs(8)**.

large_dir

This feature increases the limit on the number of files per directory by raising the maximum size of directories and, for hashed b-tree directories (see **dir_index**), the maximum height of the hashed b-tree used to store the directory entries.

large_file

This feature flag is set automatically by modern kernels when a file larger than 2 gigabytes is created. Very old kernels could not handle large files, so this feature flag was used to prohibit those kernels from mounting file systems that they could not understand.

metadata_csum

This ext4 feature enables metadata checksumming. This feature stores checksums for all of the file system metadata (superblock, group descriptor blocks, inode and block bitmaps, directories, and extent tree blocks). The checksum algorithm used for the metadata blocks is different than the one used for group descriptors with the **uninit_bg** feature. These two features are incompatible and **metadata_csum** will be used preferentially instead of **uninit_bg**.

metadata_csum_seed

This feature allows the file system to store the metadata checksum seed in the superblock, which allows the administrator to change the UUID of a file system using the **metadata_csum** feature while it is mounted.

meta_bg

This ext4 feature allows file systems to be resized on-line without explicitly needing to reserve space for growth in the size of the block group descriptors. This scheme is also used to resize file systems which are larger than 2^{32} blocks. It is not recommended that this feature be set when a file system is created, since this alternate method of storing the block group descriptors will slow down the time needed to mount the file system, and newer kernels can automatically set this feature as necessary when doing an online resize and no more reserved space is available in the resize inode.

mmp

This ext4 feature provides multiple mount protection (MMP). MMP helps to protect the file system from being multiply mounted and is useful in shared storage environments.

project

This ext4 feature provides project quota support. With this feature, the project ID of inode will be managed when the file system is mounted.

quota

Create quota inodes (inode #3 for userquota and inode #4 for group quota) and set them in the superblock. With this feature, the quotas will be enabled automatically when the file system is mounted.

Causes the quota files (i.e., user.quota and group.quota which existed in the older quota design) to be hidden inodes.

resize_inode

This file system feature indicates that space has been reserved so that the block group descriptor table can be extended while resizing a mounted file system. The online resize operation is carried out by the kernel, triggered by **resize2fs(8)**. By default **mke2fs** will attempt to reserve enough space so that the file system may grow to 1024 times its initial size. This can be changed using the **resize** extended option.

This feature requires that the **sparse_super** or **sparse_super2** feature be enabled.

sparse_super

This file system feature is set on all modern ext2, ext3, and ext4 file systems. It indicates that backup copies of the superblock and block group descriptors are present only in a few block groups, not all of them.

sparse_super2

This feature indicates that there will only be at most two backup superblocks and block group descriptors. The block groups used to store the backup superblock(s) and blockgroup descriptor(s) are stored in the superblock, but typically, one will be located at the beginning of block group #1, and one in the last block group in the file system. This feature is essentially a more extreme version of sparse_super and is designed to allow a much larger percentage of the disk to have contiguous blocks available for data files.

stable_inodes

Marks the file system's inode numbers and UUID as stable. **r esize2fs(8)** will not allow shrinking a file system with this feature, nor will **tune2fs(8)** allow changing its UUID. This feature allows the use of specialized encryption settings that make use of the inode numbers and UUID. Note that the **encrypt** feature still needs to be enabled separately. **stable_inodes** is a "compat" feature, so old kernels will allow it.

uninit_bg

This ext4 file system feature indicates that the block group descriptors will be protected using checksums, making it safe for **mke2fs(8)** to create a file system without initializing all of the block groups. The kernel will keep a high watermark of unused inodes, and initialize inode tables and blocks lazily. This feature speeds up the time to check the file system using **e2fsck(8)**, and it also speeds up the time required for **mke2fs(8)** to create the file system.

verity

Enables support for verity protected files. Verity files are readonly, and their data is transparently verified against a Merkle tree hidden past the end of the file. Using the Merkle tree's root hash, a verity file can be efficiently authenticated, independent of the file's size.

This feature is most useful for authenticating important read-only files on read-write file systems. If the file system itself is read-only, then using dm-verity to authenticate the entire block device may provide much better security.

MOUNT OPTIONS

This section describes mount options which are specific to ext2, ext3, and ext4. Other generic mount options may be used as well; see **mount(8)** for details.

Mount options for ext2

The 'ext2' file system is the standard Linux file system. Since Linux 2.5.46, for most mount options the default is determined by the file system superblock. Set them with **tune2fs(8)**.

acl|noacl

Support POSIX Access Control Lists (or not). See the **acl(5)** manual page.

bsddf|minixdf

Set the behavior for the *statfs* system call. The **minixdf** behavior is to return in the *f_blocks* field the total number of blocks of the file system, while the **bsddf** behavior (which is the default) is to subtract the overhead blocks used by the ext2 file system and not available for file storage. Thus

```
% mount /k -o minixdf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2630655	86954	2412169	3%	/k

```
% mount /k -o bsddf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2543714	13	2412169	0%	/k

(Note that this example shows that one can add command line options to the options given in */etc/fstab*.)

check=none or nocheck

No checking is done at mount time. This is the default. This is fast. It is wise to invoke **e2fsck(8)** every now and then, e.g. at boot time. The non-default behavior is unsupported (check=normal and check=strict options have been removed). Note that these mount options don't have to be supported if ext4 kernel driver is used for ext2 and ext3 file systems.

debug Print debugging info upon each (re)mount.

errors={continue|remount-ro|panic}

Define the behavior when an error is encountered. (Either ignore errors and just mark the file system erroneous and continue, or remount the file system read-only, or panic and halt the system.) The default is set in the file system superblock, and can be changed using **tune2fs(8)**.

grpuid|bsdgroups and nogrpid|sysvgroups

These options define what group id a newly created file gets. When **grpuid** is set, it takes the group id of the directory in which it is created; otherwise (the default) it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

grpquota|noquota|quota|usrquota

The **usrquota** (same as **quota**) mount option enables user quota support on the file system. **grpquota** enables group quotas support. You need the quota utilities to actually enable and manage the quota system.

nouid32

Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values.

oldalloc or orlov

Use old allocator or Orlov allocator for new inodes. Orlov is default.

resgid=n and resuid=n

The ext2 file system reserves a certain percentage of the available space (by default 5%, see **mke2fs(8)** and **tune2fs(8)**). These options determine who can use the reserved blocks. (Roughly: whoever has the specified uid, or belongs to the specified group.)

sb=n Instead of using the normal superblock, use an alternative superblock specified by *n*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system's blocksize, the number of blocks per group, and features such as **sparse_super**.

Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system's layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).

The block number here uses 1 k units. Thus, if you want to use logical block 32768 on a file system with 4 k blocks, use "sb=131072".

user_xattr|nouser_xattr

Support "user." extended attributes (or not).

Mount options for ext3

The ext3 file system is a version of the ext2 file system which has been enhanced with journaling. It supports the same options as ext2 as well as the following additions:

journal_dev=devnum/journal_path=path

When the external journal device's major/minor numbers have changed, these options allow the user to specify the new journal location. The journal device is identified either through its new major/minor numbers encoded in devnum, or via a path to the device.

norecovery/noload

Don't load the journal on mounting. Note that if the file system was not unmounted cleanly, skipping the journal replay will lead to the file system containing inconsistencies that can lead to any number of problems.

data={journal|ordered|writeback}

Specifies the journaling mode for file data. Metadata is always journaled. To use modes other than **ordered** on the root file system, pass the mode to the kernel as boot parameter, e.g. *root-flags=data=journal*.

journal

All data is committed into the journal prior to being written into the main file system.

ordered

This is the default mode. All data is forced directly out to the main file system prior to its metadata being committed to the journal.

writeback

Data ordering is not preserved – data may be written into the main file system after its metadata has been committed to the journal. This is rumoured to be the highest-throughput option. It guarantees internal file system integrity, however it can allow old data to appear in files after a crash and journal recovery.

data_err=ignore

Just print an error message if an error occurs in a file data buffer in ordered mode.

data_err=abort

Abort the journal if an error occurs in a file data buffer in ordered mode.

barrier=0 / barrier=1

This disables / enables the use of write barriers in the jbd code. barrier=0 disables, barrier=1 enables (default). This also requires an IO stack which can support barriers, and if jbd gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

commit=nrsec

Start a journal commit every *nrsec* seconds. The default value is 5 seconds. Zero means default.

user_xattr

Enable Extended User Attributes. See the **attr(5)** manual page.

jqfmt={vfsold|vfsv0|vfsv1}

Apart from the old quota system (as in ext2, jqfmt=vfsold aka version 1 quota) ext3 also supports journaled quotas (version 2 quota). jqfmt=vfsv0 or jqfmt=vfsv1 enables journaled quotas. Journaled quotas have the advantage that even after a crash no quota check is required. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

usrjquota=aquota.user|grpjquota=aquota.group

For journaled quotas (jqfmt=vfsv0 or jqfmt=vfsv1), the mount options **usrjquota=aquota.user** and **grpjquota=aquota.group** are required to tell the quota system which quota database files to use. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

Mount options for ext4

The ext4 file system is an advanced level of the ext3 file system which incorporates scalability and reliability enhancements for supporting large file system.

The options **journal_dev**, **journal_path**, **norecovery**, **noload**, **data**, **commit**, **orlov**, **oldalloc**,

[no]user_xattr, [no]acl, bsddf, minixdf, debug, errors, data_err, grpid, bsdgroups, nogrpid, sysv-groups, resgid, resuid, sb, quota, noquota, nouid32, grpquota, usrquota, usrjquota, grpquota, and jqfmt are backwardly compatible with ext3 or ext2.

journal_checksum | nojournal_checksum

The journal_checksum option enables checksumming of the journal transactions. This will allow the recovery code in e2fsck and the kernel to detect corruption in the kernel. It is a compatible change and will be ignored by older kernels.

journal_async_commit

Commit block can be written to disk without waiting for descriptor blocks. If enabled older kernels cannot mount the device. This will enable 'journal_checksum' internally.

barrier=0 / barrier=1 / barrier / nobarrier

These mount options have the same effect as in ext3. The mount options "barrier" and "nobarrier" are added for consistency with other ext4 mount options.

The ext4 file system enables write barriers by default.

inode_readahead_blks=n

This tuning parameter controls the maximum number of inode table blocks that ext4's inode table readahead algorithm will pre-read into the buffer cache. The value must be a power of 2. The default value is 32 blocks.

stripe=n

Number of file system blocks that mballoc will try to use for allocation size and alignment. For RAID5/6 systems this should be the number of data disks * RAID chunk size in file system blocks.

delalloc

Deferring block allocation until write-out time.

nodealloc

Disable delayed allocation. Blocks are allocated when data is copied from user to page cache.

max_batch_time=usec

Maximum amount of time ext4 should wait for additional file system operations to be batch together with a synchronous write operation. Since a synchronous write operation is going to force a commit and then a wait for the I/O complete, it doesn't cost much, and can be a huge throughput win, we wait for a small amount of time to see if any other transactions can piggyback on the synchronous write. The algorithm used is designed to automatically tune for the speed of the disk, by measuring the amount of time (on average) that it takes to finish committing a transaction. Call this time the "commit time". If the time that the transaction has been running is less than the commit time, ext4 will try sleeping for the commit time to see if other operations will join the transaction. The commit time is capped by the max_batch_time, which defaults to 15000 µs (15 ms). This optimization can be turned off entirely by setting max_batch_time to 0.

min_batch_time=usec

This parameter sets the commit time (as described above) to be at least min_batch_time. It defaults to zero microseconds. Increasing this parameter may improve the throughput of multi-threaded, synchronous workloads on very fast disks, at the cost of increasing latency.

journal_ioprio=prio

The I/O priority (from 0 to 7, where 0 is the highest priority) which should be used for I/O operations submitted by kjournald2 during a commit operation. This defaults to 3, which is a slightly higher priority than the default I/O priority.

abort Simulate the effects of calling ext4_abort() for debugging purposes. This is normally used while remounting a file system which is already mounted.

auto_da_alloc|noauto_da_alloc

Many broken applications don't use fsync() when replacing existing files via patterns such as

```
fd = open("foo.new")/write(fd,...)/close(fd)/ rename("foo.new", "foo")
```

or worse yet

```
fd = open("foo", O_TRUNC)/write(fd,...)/close(fd).
```

If auto_da_alloc is enabled, ext4 will detect the replace-via-rename and replace-via-truncate patterns and force that any delayed allocation blocks are allocated such that at the next journal commit, in the default data=ordered mode, the data blocks of the new file are forced to disk before the rename() operation is committed. This provides roughly the same level of guarantees as ext3, and avoids the "zero-length" problem that can happen when a system crashes before the delayed allocation blocks are forced to disk.

noinit_itable

Do not initialize any uninitialized inode table blocks in the background. This feature may be used by installation CD's so that the install process can complete as quickly as possible; the inode table initialization process would then be deferred until the next time the file system is mounted.

init_itable=n

The lazy itable init code will wait n times the number of milliseconds it took to zero out the previous block group's inode table. This minimizes the impact on system performance while the file system's inode table is being initialized.

discard/nodiscard

Controls whether ext4 should issue discard/TRIM commands to the underlying block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs, but it is off by default until sufficient testing has been done.

block_validity/noblock_validity

This option enables/disables the in-kernel facility for tracking file system metadata blocks within internal data structures. This allows multi-block allocator and other routines to quickly locate extents which might overlap with file system metadata blocks. This option is intended for debugging purposes and since it negatively affects the performance, it is off by default.

dioread_lock/dioread_nolock

Controls whether or not ext4 should use the DIO read locking. If the dioread_nolock option is specified ext4 will allocate uninitialized extent before buffer write and convert the extent to initialized after IO completes. This approach allows ext4 code to avoid using inode mutex, which improves scalability on high speed storages. However this does not work with data journaling and dioread_nolock option will be ignored with kernel warning. Note that dioread_nolock code path is only used for extent-based files. Because of the restrictions this options comprises it is off by default (e.g. dioread_lock).

max_dir_size_kb=n

This limits the size of the directories so that any attempt to expand them beyond the specified limit in kilobytes will cause an ENOSPC error. This is useful in memory-constrained environments, where a very large directory can cause severe performance problems or even provoke the Out Of Memory killer. (For example, if there is only 512 MB memory available, a 176 MB directory may seriously cramp the system's style.)

i_version

Enable 64-bit inode version support. This option is off by default.

nombcache

This option disables use of mbcache for extended attribute deduplication. On systems where extended attributes are rarely or never shared between files, use of mbcache for deduplication adds

unnecessary computational overhead.

prjquota

The prjquota mount option enables project quota support on the file system. You need the quota utilities to actually enable and manage the quota system. This mount option requires the **project** file system feature.

FILE ATTRIBUTES

The ext2, ext3, and ext4 file systems support setting the following file attributes on Linux systems using the **chattr(1)** utility:

a - append only

A - no atime updates

d - no dump

D - synchronous directory updates

i - immutable

S - synchronous updates

u - undeletable

In addition, the ext3 and ext4 file systems support the following flag:

j - data journaling

Finally, the ext4 file system also supports the following flag:

e - extents format

For descriptions of these attribute flags, please refer to the **chattr(1)** man page.

KERNEL SUPPORT

This section lists the file system driver (e.g., ext2, ext3, ext4) and upstream kernel version where a particular file system feature was supported. Note that in some cases the feature was present in earlier kernel versions, but there were known, serious bugs. In other cases the feature may still be considered in an experimental state. Finally, note that some distributions may have backported features into older kernels; in particular the kernel versions in certain "enterprise distributions" can be extremely misleading.

filetype	ext2, 2.2.0
sparse_super	ext2, 2.2.0
large_file	ext2, 2.2.0
has_journal	ext3, 2.4.15
ext_attr	ext2/ext3, 2.6.0
dir_index	ext3, 2.6.0
resize_inode	ext3, 2.6.10 (online resizing)
64bit	ext4, 2.6.28
dir_nlink	ext4, 2.6.28

extent	ext4, 2.6.28
extra_isize	ext4, 2.6.28
flex_bg	ext4, 2.6.28
huge_file	ext4, 2.6.28
meta_bg	ext4, 2.6.28
uninit_bg	ext4, 2.6.28
mmp	ext4, 3.0
bigalloc	ext4, 3.2
quota	ext4, 3.6
inline_data	ext4, 3.8
sparse_super2	ext4, 3.16
metadata_csum	ext4, 3.18
encrypt	ext4, 4.1
metadata_csum_seed	ext4, 4.4
project	ext4, 4.5
ea_inode	ext4, 4.13
large_dir	ext4, 4.13
casefold	ext4, 5.2
verity	ext4, 5.4
stable_inodes	ext4, 5.5

SEE ALSO

mke2fs(8), mke2fs.conf(5), e2fsck(8), dumpe2fs(8), tune2fs(8), debugfs(8), mount(8), chattr(1)

NAME

ext2 – the second extended file system
ext3 – the third extended file system
ext4 – the fourth extended file system

DESCRIPTION

The second, third, and fourth extended file systems, or ext2, ext3, and ext4 as they are commonly known, are Linux file systems that have historically been the default file system for many Linux distributions. They are general purpose file systems that have been designed for extensibility and backwards compatibility. In particular, file systems previously intended for use with the ext2 and ext3 file systems can be mounted using the ext4 file system driver, and indeed in many modern Linux distributions, the ext4 file system driver has been configured to handle mount requests for ext2 and ext3 file systems.

FILE SYSTEM FEATURES

A file system formatted for ext2, ext3, or ext4 can have some collection of the following file system feature flags enabled. Some of these features are not supported by all implementations of the ext2, ext3, and ext4 file system drivers, depending on Linux kernel version in use. On other operating systems, such as the GNU/HURD or FreeBSD, only a very restrictive set of file system features may be supported in their implementations of ext2.

64bit

Enables the file system to be larger than 2^{32} blocks. This feature is set automatically, as needed, but it can be useful to specify this feature explicitly if the file system might need to be resized larger than 2^{32} blocks, even if it was smaller than that threshold when it was originally created. Note that some older kernels and older versions of e2fsprogs will not support file systems with this ext4 feature enabled.

bigalloc

This ext4 feature enables clustered block allocation, so that the unit of allocation is a power of two number of blocks. That is, each bit in the what had traditionally been known as the block allocation bitmap now indicates whether a cluster is in use or not, where a cluster is by default composed of 16 blocks. This feature can decrease the time spent on doing block allocation and brings smaller fragmentation, especially for large files. The size can be specified using the **mke2fs -C** option.

Warning: The bigalloc feature is still under development, and may not be fully supported with your kernel or may have various bugs. Please see the web page <http://ext4.wiki.kernel.org/index.php/Bigalloc> for details. May clash with delayed allocation (see **nodelalloc** mount option).

This feature requires that the **extent** feature be enabled.

casifold

This ext4 feature provides file system level character encoding support for directories with the casifold (+F) flag enabled. This feature is name-preserving on the disk, but it allows applications to lookup for a file in the file system using an encoding equivalent version of the file name.

dir_index

Use hashed b-trees to speed up name lookups in large directories. This feature is supported by ext3 and ext4 file systems, and is ignored by ext2 file systems.

dir_nlink

Normally, ext4 allows an inode to have no more than 65,000 hard links. This applies to regular files as well as directories, which means that there can be no more than 64,998 subdirectories in a directory (because each of the '.' and '..' entries, as well as the directory entry for the directory in its parent directory counts as a hard link). This feature lifts this limit by causing ext4 to use a link count of 1 to indicate that the number of hard links to a directory is not known when the link count might exceed the maximum count limit.

ea_inode

Normally, a file's extended attributes and associated metadata must fit within the inode or the inode's associated extended attribute block. This feature allows the value of each extended attribute to be placed in the data blocks of a separate inode if necessary, increasing the limit on the size and number of extended attributes per file.

encrypt

Enables support for file-system level encryption of data blocks and file names. The inode metadata (timestamps, file size, user/group ownership, etc.) is *not* encrypted.

This feature is most useful on file systems with multiple users, or where not all files should be encrypted. In many use cases, especially on single-user systems, encryption at the block device layer using dm-crypt may provide much better security.

ext_attr

This feature enables the use of extended attributes. This feature is supported by ext2, ext3, and ext4.

extent

This ext4 feature allows the mapping of logical block numbers for a particular inode to physical blocks on the storage device to be stored using an extent tree, which is a more efficient data structure than the traditional indirect block scheme used by the ext2 and ext3 file systems. The use of the extent tree decreases metadata block overhead, improves file system performance, and decreases the need to run **e2fsck(8)** on the file system. (Note: both **extent** and **extents** are accepted as valid names for this feature for historical/backwards compatibility reasons.)

extra_isize

This ext4 feature reserves a specific amount of space in each inode for extended metadata such as nanosecond timestamps and file creation time, even if the current kernel does not currently need to reserve this much space. Without this feature, the kernel will reserve the amount of space for features it currently needs, and the rest may be consumed by extended attributes.

For this feature to be useful the inode size must be 256 bytes in size or larger.

filetype

This feature enables the storage of file type information in directory entries. This feature is supported by ext2, ext3, and ext4.

flex_bg

This ext4 feature allows the per-block group metadata (allocation bitmaps and inode tables) to be placed anywhere on the storage media. In addition, **mke2fs** will place the per-block group metadata together starting at the first block group of each "flex_bg group". The size of the **flex_bg** group can be specified using the **-G** option.

has_journal

Create a journal to ensure file system consistency even across unclean shutdowns. Setting the file system feature is equivalent to using the **-j** option with **mke2fs** or **tune2fs**. This feature is supported by ext3 and ext4, and ignored by the ext2 file system driver.

huge_file

This ext4 feature allows files to be larger than 2 terabytes in size.

inline_data

Allow data to be stored in the inode and extended attribute area.

journal_dev

This feature is enabled on the superblock found on an external journal device. The block size for the external journal must be the same as the file system which uses it.

The external journal device can be used by a file system by specifying the **-J device=<external-device>** option to **mke2fs(8)** or **tune2fs(8)**.

large_dir

This feature increases the limit on the number of files per directory by raising the maximum size of directories and, for hashed b-tree directories (see **dir_index**), the maximum height of the hashed b-tree used to store the directory entries.

large_file

This feature flag is set automatically by modern kernels when a file larger than 2 gigabytes is created. Very old kernels could not handle large files, so this feature flag was used to prohibit those kernels from mounting file systems that they could not understand.

metadata_csum

This ext4 feature enables metadata checksumming. This feature stores checksums for all of the file system metadata (superblock, group descriptor blocks, inode and block bitmaps, directories, and extent tree blocks). The checksum algorithm used for the metadata blocks is different than the one used for group descriptors with the **uninit_bg** feature. These two features are incompatible and **metadata_csum** will be used preferentially instead of **uninit_bg**.

metadata_csum_seed

This feature allows the file system to store the metadata checksum seed in the superblock, which allows the administrator to change the UUID of a file system using the **metadata_csum** feature while it is mounted.

meta_bg

This ext4 feature allows file systems to be resized on-line without explicitly needing to reserve space for growth in the size of the block group descriptors. This scheme is also used to resize file systems which are larger than 2^{32} blocks. It is not recommended that this feature be set when a file system is created, since this alternate method of storing the block group descriptors will slow down the time needed to mount the file system, and newer kernels can automatically set this feature as necessary when doing an online resize and no more reserved space is available in the resize inode.

mmp

This ext4 feature provides multiple mount protection (MMP). MMP helps to protect the file system from being multiply mounted and is useful in shared storage environments.

project

This ext4 feature provides project quota support. With this feature, the project ID of inode will be managed when the file system is mounted.

quota

Create quota inodes (inode #3 for userquota and inode #4 for group quota) and set them in the superblock. With this feature, the quotas will be enabled automatically when the file system is mounted.

Causes the quota files (i.e., user.quota and group.quota which existed in the older quota design) to be hidden inodes.

resize_inode

This file system feature indicates that space has been reserved so that the block group descriptor table can be extended while resizing a mounted file system. The online resize operation is carried out by the kernel, triggered by **resize2fs(8)**. By default **mke2fs** will attempt to reserve enough space so that the file system may grow to 1024 times its initial size. This can be changed using the **resize** extended option.

This feature requires that the **sparse_super** or **sparse_super2** feature be enabled.

sparse_super

This file system feature is set on all modern ext2, ext3, and ext4 file systems. It indicates that backup copies of the superblock and block group descriptors are present only in a few block groups, not all of them.

sparse_super2

This feature indicates that there will only be at most two backup superblocks and block group descriptors. The block groups used to store the backup superblock(s) and blockgroup descriptor(s) are stored in the superblock, but typically, one will be located at the beginning of block group #1, and one in the last block group in the file system. This feature is essentially a more extreme version of sparse_super and is designed to allow a much larger percentage of the disk to have contiguous blocks available for data files.

stable_inodes

Marks the file system's inode numbers and UUID as stable. **r esize2fs(8)** will not allow shrinking a file system with this feature, nor will **tune2fs(8)** allow changing its UUID. This feature allows the use of specialized encryption settings that make use of the inode numbers and UUID. Note that the **encrypt** feature still needs to be enabled separately. **stable_inodes** is a "compat" feature, so old kernels will allow it.

uninit_bg

This ext4 file system feature indicates that the block group descriptors will be protected using checksums, making it safe for **mke2fs(8)** to create a file system without initializing all of the block groups. The kernel will keep a high watermark of unused inodes, and initialize inode tables and blocks lazily. This feature speeds up the time to check the file system using **e2fsck(8)**, and it also speeds up the time required for **mke2fs(8)** to create the file system.

verity

Enables support for verity protected files. Verity files are readonly, and their data is transparently verified against a Merkle tree hidden past the end of the file. Using the Merkle tree's root hash, a verity file can be efficiently authenticated, independent of the file's size.

This feature is most useful for authenticating important read-only files on read-write file systems. If the file system itself is read-only, then using dm-verity to authenticate the entire block device may provide much better security.

MOUNT OPTIONS

This section describes mount options which are specific to ext2, ext3, and ext4. Other generic mount options may be used as well; see **mount(8)** for details.

Mount options for ext2

The 'ext2' file system is the standard Linux file system. Since Linux 2.5.46, for most mount options the default is determined by the file system superblock. Set them with **tune2fs(8)**.

acl|noacl

Support POSIX Access Control Lists (or not). See the **acl(5)** manual page.

bsddf|minixdf

Set the behavior for the *statfs* system call. The **minixdf** behavior is to return in the *f_blocks* field the total number of blocks of the file system, while the **bsddf** behavior (which is the default) is to subtract the overhead blocks used by the ext2 file system and not available for file storage. Thus

```
% mount /k -o minixdf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2630655	86954	2412169	3%	/k

```
% mount /k -o bsddf; df /k; umount /k
```

File System	1024-blocks	Used	Available	Capacity	Mounted on
/dev/sda6	2543714	13	2412169	0%	/k

(Note that this example shows that one can add command line options to the options given in */etc/fstab*.)

check=none or nocheck

No checking is done at mount time. This is the default. This is fast. It is wise to invoke **e2fsck(8)** every now and then, e.g. at boot time. The non-default behavior is unsupported (check=normal and check=strict options have been removed). Note that these mount options don't have to be supported if ext4 kernel driver is used for ext2 and ext3 file systems.

debug Print debugging info upon each (re)mount.

errors={continue|remount-ro|panic}

Define the behavior when an error is encountered. (Either ignore errors and just mark the file system erroneous and continue, or remount the file system read-only, or panic and halt the system.) The default is set in the file system superblock, and can be changed using **tune2fs(8)**.

grpid|bsdgroups and nogrpid|sysvgroups

These options define what group id a newly created file gets. When **grpid** is set, it takes the group id of the directory in which it is created; otherwise (the default) it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

grpquota|noquota|quota|usrquota

The **usrquota** (same as **quota**) mount option enables user quota support on the file system. **grpquota** enables group quotas support. You need the quota utilities to actually enable and manage the quota system.

nouid32

Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values.

oldalloc or orlov

Use old allocator or Orlov allocator for new inodes. Orlov is default.

resgid=n and resuid=n

The ext2 file system reserves a certain percentage of the available space (by default 5%, see **mke2fs(8)** and **tune2fs(8)**). These options determine who can use the reserved blocks. (Roughly: whoever has the specified uid, or belongs to the specified group.)

sb=n Instead of using the normal superblock, use an alternative superblock specified by *n*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system's blocksize, the number of blocks per group, and features such as **sparse_super**.

Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system's layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).

The block number here uses 1 k units. Thus, if you want to use logical block 32768 on a file system with 4 k blocks, use "sb=131072".

user_xattr|nouser_xattr

Support "user." extended attributes (or not).

Mount options for ext3

The ext3 file system is a version of the ext2 file system which has been enhanced with journaling. It supports the same options as ext2 as well as the following additions:

journal_dev=devnum/journal_path=path

When the external journal device's major/minor numbers have changed, these options allow the user to specify the new journal location. The journal device is identified either through its new major/minor numbers encoded in devnum, or via a path to the device.

norecovery/noload

Don't load the journal on mounting. Note that if the file system was not unmounted cleanly, skipping the journal replay will lead to the file system containing inconsistencies that can lead to any number of problems.

data={journal|ordered|writeback}

Specifies the journaling mode for file data. Metadata is always journaled. To use modes other than **ordered** on the root file system, pass the mode to the kernel as boot parameter, e.g. *root-flags=data=journal*.

journal

All data is committed into the journal prior to being written into the main file system.

ordered

This is the default mode. All data is forced directly out to the main file system prior to its metadata being committed to the journal.

writeback

Data ordering is not preserved – data may be written into the main file system after its metadata has been committed to the journal. This is rumoured to be the highest-throughput option. It guarantees internal file system integrity, however it can allow old data to appear in files after a crash and journal recovery.

data_err=ignore

Just print an error message if an error occurs in a file data buffer in ordered mode.

data_err=abort

Abort the journal if an error occurs in a file data buffer in ordered mode.

barrier=0 / barrier=1

This disables / enables the use of write barriers in the jbd code. barrier=0 disables, barrier=1 enables (default). This also requires an IO stack which can support barriers, and if jbd gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

commit=nrsec

Start a journal commit every *nrsec* seconds. The default value is 5 seconds. Zero means default.

user_xattr

Enable Extended User Attributes. See the **attr(5)** manual page.

jqfmt={vfsold|vfsv0|vfsv1}

Apart from the old quota system (as in ext2, jqfmt=vfsold aka version 1 quota) ext3 also supports journaled quotas (version 2 quota). jqfmt=vfsv0 or jqfmt=vfsv1 enables journaled quotas. Journaled quotas have the advantage that even after a crash no quota check is required. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

usrjquota=aquota.user|grpjquota=aquota.group

For journaled quotas (jqfmt=vfsv0 or jqfmt=vfsv1), the mount options **usrjquota=aquota.user** and **grpjquota=aquota.group** are required to tell the quota system which quota database files to use. When the **quota** file system feature is enabled, journaled quotas are used automatically, and this mount option is ignored.

Mount options for ext4

The ext4 file system is an advanced level of the ext3 file system which incorporates scalability and reliability enhancements for supporting large file system.

The options **journal_dev**, **journal_path**, **norecovery**, **noload**, **data**, **commit**, **orlov**, **oldalloc**,

[no]user_xattr, [no]acl, bsddf, minixdf, debug, errors, data_err, grpid, bsdgroups, nogrpid, sysv-groups, resgid, resuid, sb, quota, noquota, nouid32, grpquota, usrquota, usrjquota, grpquota, and jqfmt are backwardly compatible with ext3 or ext2.

journal_checksum | nojournal_checksum

The journal_checksum option enables checksumming of the journal transactions. This will allow the recovery code in e2fsck and the kernel to detect corruption in the kernel. It is a compatible change and will be ignored by older kernels.

journal_async_commit

Commit block can be written to disk without waiting for descriptor blocks. If enabled older kernels cannot mount the device. This will enable 'journal_checksum' internally.

barrier=0 / barrier=1 / barrier / nobarrier

These mount options have the same effect as in ext3. The mount options "barrier" and "nobarrier" are added for consistency with other ext4 mount options.

The ext4 file system enables write barriers by default.

inode_readahead_blks=n

This tuning parameter controls the maximum number of inode table blocks that ext4's inode table readahead algorithm will pre-read into the buffer cache. The value must be a power of 2. The default value is 32 blocks.

stripe=n

Number of file system blocks that mballoc will try to use for allocation size and alignment. For RAID5/6 systems this should be the number of data disks * RAID chunk size in file system blocks.

delalloc

Deferring block allocation until write-out time.

nodealloc

Disable delayed allocation. Blocks are allocated when data is copied from user to page cache.

max_batch_time=usec

Maximum amount of time ext4 should wait for additional file system operations to be batch together with a synchronous write operation. Since a synchronous write operation is going to force a commit and then a wait for the I/O complete, it doesn't cost much, and can be a huge throughput win, we wait for a small amount of time to see if any other transactions can piggyback on the synchronous write. The algorithm used is designed to automatically tune for the speed of the disk, by measuring the amount of time (on average) that it takes to finish committing a transaction. Call this time the "commit time". If the time that the transaction has been running is less than the commit time, ext4 will try sleeping for the commit time to see if other operations will join the transaction. The commit time is capped by the max_batch_time, which defaults to 15000 µs (15 ms). This optimization can be turned off entirely by setting max_batch_time to 0.

min_batch_time=usec

This parameter sets the commit time (as described above) to be at least min_batch_time. It defaults to zero microseconds. Increasing this parameter may improve the throughput of multi-threaded, synchronous workloads on very fast disks, at the cost of increasing latency.

journal_ioprio=prio

The I/O priority (from 0 to 7, where 0 is the highest priority) which should be used for I/O operations submitted by kjournald2 during a commit operation. This defaults to 3, which is a slightly higher priority than the default I/O priority.

abort Simulate the effects of calling ext4_abort() for debugging purposes. This is normally used while remounting a file system which is already mounted.

auto_da_alloc|noauto_da_alloc

Many broken applications don't use fsync() when replacing existing files via patterns such as

```
fd = open("foo.new")/write(fd,...)/close(fd)/ rename("foo.new", "foo")
```

or worse yet

```
fd = open("foo", O_TRUNC)/write(fd,...)/close(fd).
```

If auto_da_alloc is enabled, ext4 will detect the replace-via-rename and replace-via-truncate patterns and force that any delayed allocation blocks are allocated such that at the next journal commit, in the default data=ordered mode, the data blocks of the new file are forced to disk before the rename() operation is committed. This provides roughly the same level of guarantees as ext3, and avoids the "zero-length" problem that can happen when a system crashes before the delayed allocation blocks are forced to disk.

noinit_itable

Do not initialize any uninitialized inode table blocks in the background. This feature may be used by installation CD's so that the install process can complete as quickly as possible; the inode table initialization process would then be deferred until the next time the file system is mounted.

init_itable=n

The lazy itable init code will wait n times the number of milliseconds it took to zero out the previous block group's inode table. This minimizes the impact on system performance while the file system's inode table is being initialized.

discard/nodiscard

Controls whether ext4 should issue discard/TRIM commands to the underlying block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs, but it is off by default until sufficient testing has been done.

block_validity/noblock_validity

This option enables/disables the in-kernel facility for tracking file system metadata blocks within internal data structures. This allows multi-block allocator and other routines to quickly locate extents which might overlap with file system metadata blocks. This option is intended for debugging purposes and since it negatively affects the performance, it is off by default.

dioread_lock/dioread_nolock

Controls whether or not ext4 should use the DIO read locking. If the dioread_nolock option is specified ext4 will allocate uninitialized extent before buffer write and convert the extent to initialized after IO completes. This approach allows ext4 code to avoid using inode mutex, which improves scalability on high speed storages. However this does not work with data journaling and dioread_nolock option will be ignored with kernel warning. Note that dioread_nolock code path is only used for extent-based files. Because of the restrictions this options comprises it is off by default (e.g. dioread_lock).

max_dir_size_kb=n

This limits the size of the directories so that any attempt to expand them beyond the specified limit in kilobytes will cause an ENOSPC error. This is useful in memory-constrained environments, where a very large directory can cause severe performance problems or even provoke the Out Of Memory killer. (For example, if there is only 512 MB memory available, a 176 MB directory may seriously cramp the system's style.)

i_version

Enable 64-bit inode version support. This option is off by default.

nombcache

This option disables use of mbcache for extended attribute deduplication. On systems where extended attributes are rarely or never shared between files, use of mbcache for deduplication adds

unnecessary computational overhead.

prjquota

The prjquota mount option enables project quota support on the file system. You need the quota utilities to actually enable and manage the quota system. This mount option requires the **project** file system feature.

FILE ATTRIBUTES

The ext2, ext3, and ext4 file systems support setting the following file attributes on Linux systems using the **chattr(1)** utility:

a - append only

A - no atime updates

d - no dump

D - synchronous directory updates

i - immutable

S - synchronous updates

u - undeletable

In addition, the ext3 and ext4 file systems support the following flag:

j - data journaling

Finally, the ext4 file system also supports the following flag:

e - extents format

For descriptions of these attribute flags, please refer to the **chattr(1)** man page.

KERNEL SUPPORT

This section lists the file system driver (e.g., ext2, ext3, ext4) and upstream kernel version where a particular file system feature was supported. Note that in some cases the feature was present in earlier kernel versions, but there were known, serious bugs. In other cases the feature may still be considered in an experimental state. Finally, note that some distributions may have backported features into older kernels; in particular the kernel versions in certain "enterprise distributions" can be extremely misleading.

filetype	ext2, 2.2.0
sparse_super	ext2, 2.2.0
large_file	ext2, 2.2.0
has_journal	ext3, 2.4.15
ext_attr	ext2/ext3, 2.6.0
dir_index	ext3, 2.6.0
resize_inode	ext3, 2.6.10 (online resizing)
64bit	ext4, 2.6.28
dir_nlink	ext4, 2.6.28

extent	ext4, 2.6.28
extra_isize	ext4, 2.6.28
flex_bg	ext4, 2.6.28
huge_file	ext4, 2.6.28
meta_bg	ext4, 2.6.28
uninit_bg	ext4, 2.6.28
mmp	ext4, 3.0
bigalloc	ext4, 3.2
quota	ext4, 3.6
inline_data	ext4, 3.8
sparse_super2	ext4, 3.16
metadata_csum	ext4, 3.18
encrypt	ext4, 4.1
metadata_csum_seed	ext4, 4.4
project	ext4, 4.5
ea_inode	ext4, 4.13
large_dir	ext4, 4.13
casefold	ext4, 5.2
verity	ext4, 5.4
stable_inodes	ext4, 5.5

SEE ALSO

mke2fs(8), mke2fs.conf(5), e2fsck(8), dumpe2fs(8), tune2fs(8), debugfs(8), mount(8), chattr(1)

NAME

extlinux – install the SYSLINUX bootloader on an ext2/ext3/ext4/btrfs/xfs filesystem

SYNOPSIS

extlinux [*options*] *directory*

DESCRIPTION

EXTLINUX is a new syslinux derivative, which boots from a Linux ext2/ext3/ext4/btrfs or xfs filesystem. It works the same way as **SYSLINUX**, with a few slight modifications. It is intended to simplify first-time installation of Linux, and for creation of rescue and other special-purpose boot disks.

The installer is designed to be run on a mounted directory. For example, if you have an ext2, ext3, ext4, or btrfs usb key mounted on /mnt, you can run the following command:

```
extlinux --install /mnt
```

OPTIONS

-H, --heads=#

Force the number of heads.

-i, --install

Install over the current bootsector.

-O, --clear-once

Clear the boot-once command.

-o, --once=*command*

Execute a command once upon boot.

-M, --menu-save=*label*

Set the label to select as default on the next boot

-r, --raid

Fall back to the next device on boot failure.

--reset-adv

Reset auxiliary data.

-S, --sectors=#

Force the number of sectors per track.

-U, --update

Updates a previous **EXTLINUX** installation.

-z, --zip

Force zipdrive geometry (-H 64 -S 32).

--device=*devicename*

Override the automatic detection of device names. This option is intended for special environments only and should not be used by normal users. Misuse of this option can cause disk corruption and lost data.

FILES

The extlinux configuration file needs to be named syslinux.cfg or extlinux.conf and needs to be stored in the extlinux installation directory. For more information about the contents of extlinux.conf, see syslinux(1) manpage, section files.

LIMITATIONS

Booting from XFS only works when an MBR partition table is used (not on GPT partition tables).

BUGS

I would appreciate hearing of any problems you have with SYSLINUX. I would also like to hear from you if you have successfully used SYSLINUX, especially if you are using it for a distribution.

If you are reporting problems, please include all possible information about your system and your BIOS; the vast majority of all problems reported turn out to be BIOS or hardware bugs, and I need as much

information as possible in order to diagnose the problems.

There is a mailing list `<syslinux@zytor.com>` for discussion among SYSLINUX users and for announcements of new and test versions. You can subscribe to this mailing list `<http://www.zytor.com/mailman/listinfo/syslinux>`.

SEE ALSO

`syslinux(1)`

NAME

fatlabel – set or get MS-DOS filesystem label or volume ID

SYNOPSIS

fatlabel [*OPTIONS*] *DEVICE* [*NEW*]

DESCRIPTION

fatlabel will display or change the volume label or volume ID on the MS-DOS filesystem located on *DEVICE*. By default it works in label mode. It can be switched to volume ID mode with the option **-i** or **--volume-id**.

If *NEW* is omitted, then the existing label or volume ID is written to the standard output. A label can't be longer than 11 bytes and should be in all upper case for best compatibility. An empty string or a label consisting only of white space is not allowed. A volume ID must be given as a hexadecimal number (no leading "0x" or similar) and must fit into 32 bits.

OPTIONS

-i, --volume-id

Switch to volume ID mode.

-r, --reset

Remove label in label mode or generate new ID in volume ID mode.

-c PAGE, --codepage=PAGE

Use DOS codepage *PAGE* to encode/decode label. By default codepage 850 is used.

-h, --help

Display a help message and terminate.

-V, --version

Show version number and terminate.

COMPATIBILITY and BUGS

For historic reasons FAT label is stored in two different locations: in the boot sector and as a special volume label entry in the root directory. MS-DOS 5.00, MS-DOS 6.22, MS-DOS 7.10, Windows 98, Windows XP and also Windows 10 read FAT label only from the root directory. Absence of the volume label in the root directory is interpreted as empty or none label, even if boot sector contains some valid label.

When Windows XP or Windows 10 system changes a FAT label it stores it only in the root directory — letting boot sector unchanged. Which leads to problems when a label is removed on Windows. Old label is still stored in the boot sector but is removed from the root directory.

dosfslabel prior to the version 3.0.7 operated only with FAT labels stored in the boot sector, completely ignoring a volume label in the root directory.

dosfslabel in versions 3.0.7–3.0.15 reads FAT labels from the root directory and in case of absence, it fall-backs to a label stored in the boot sector. Change operation resulted in updating a label in the boot sector and *sometimes* also in the root directory due to the bug. That bug was fixed in **dosfslabel** version 3.0.16 and since this version **dosfslabel** updates label in both location.

Since version 4.2, **fatlabel** reads a FAT label only from the root directory (like MS-DOS and Windows systems), but changes a FAT label in both locations. In version 4.2 was fixed handling of empty labels and labels which starts with a byte 0xE5. Also in this version was added support for non-ASCII labels according to the specified DOS codepage and were added checks if a new label is valid.

It is strongly suggested to not use **dosfslabel** prior to version 3.0.16.

DOS CODEPAGES

MS-DOS and Windows systems use DOS (OEM) codepage for encoding and decoding FAT label. In Windows systems DOS codepage is global for all running applications and cannot be configured explicitly. It is

set implicitly by option *Language for non-Unicode programs* available in *Regional and Language Options* via *Control Panel*. Default DOS codepage for fatlabel is 850. See following mapping table between DOS codepage and Language for non-Unicode programs:

Codepage	Language
437	English (India), English (Malaysia), English (Republic of the Philippines), English (Singapore), English (South Africa), English (United States), English (Zimbabwe), Filipino, Hausa, Igbo, Inuktitut, Kinyarwanda, Kiswahili, Yoruba
720	Arabic, Dari, Persian, Urdu, Uyghur
737	Greek
775	Estonian, Latvian, Lithuanian
850	Afrikaans, Alsatian, Basque, Breton, Catalan, Corsican, Danish, Dutch, English (Australia), English (Belize), English (Canada), English (Caribbean), English (Ireland), English (Jamaica), English (New Zealand), English (Trinidad and Tobago), English (United Kingdom), Faroese, Finnish, French, Frisian, Galician, German, Greenlandic, Icelandic, Indonesian, Irish, isiXhosa, isiZulu, Italian, K'iche, Lower Sorbian, Luxembourgish, Malay, Mapudungun, Mohawk, Norwegian, Occitan, Portuguese, Quechua, Romansh, Sami, Scottish Gaelic, Sesotho sa Leboa, Setswana, Spanish, Swedish, Tamazight, Upper Sorbian, Welsh, Wolof
852	Albanian, Bosnian (Latin), Croatian, Czech, Hungarian, Polish, Romanian, Serbian (Latin), Slovak, Slovenian, Turkmen
855	Bosnian (Cyrillic), Serbian (Cyrillic)
857	Azeri (Latin), Turkish, Uzbek (Latin)
862	Hebrew
866	Azeri (Cyrillic), Bashkir, Belarusian, Bulgarian, Kyrgyz, Macedonian, Mongolian, Russian, Tajik, Tatar, Ukrainian, Uzbek (Cyrillic), Yakut
874	Thai
932	Japanese
936	Chinese (Simplified)
949	Korean
950	Chinese (Traditional)
1258	Vietnamese

SEE ALSO

fsck.fat(8), mkfs.fat(8)

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

fdisk – manipulate disk partition table

SYNOPSIS

fdisk [options] *device*

fdisk -l [*device...*]

DESCRIPTION

fdisk is a dialog–driven program for creation and manipulation of partition tables. It understands GPT, MBR, Sun, SGI and BSD partition tables.

Block devices can be divided into one or more logical disks called *partitions*. This division is recorded in the *partition table*, usually found in sector 0 of the disk. (In the BSD world one talks about ‘disk slices’ and a ‘disklabel’.)

All partitioning is driven by device I/O limits (the topology) by default. **fdisk** is able to optimize the disk layout for a 4K–sector size and use an alignment offset on modern devices for MBR and GPT. It is always a good idea to follow **fdisk**’s defaults as the default values (e.g., first and last partition sectors) and partition sizes specified by the +/−<size>{M,G,...} notation are always aligned according to the device properties.

CHS (Cylinder–Head–Sector) addressing is deprecated and not used by default. Please, do not follow old articles and recommendations with **fdisk -S <n> -H <n>** advices for SSD or 4K–sector devices.

Note that **partx(8)** provides a rich interface for scripts to print disk layouts, **fdisk** is mostly designed for humans. Backward compatibility in the output of **fdisk** is not guaranteed. The input (the commands) should always be backward compatible.

OPTIONS

-b, --sector-size *sectorsize*

Specify the sector size of the disk. Valid values are 512, 1024, 2048, and 4096. (Recent kernels know the sector size. Use this option only on old kernels or to override the kernel’s ideas.) Since util–linux–2.17, **fdisk** differentiates between logical and physical sector size. This option changes both sector sizes to *sectorsize*.

-B, --protect-boot

Don’t erase the beginning of the first disk sector when creating a new disk label. This feature is supported for GPT and MBR.

-c, --compatibility[=mode]

Specify the compatibility mode, ‘dos’ or ‘nondos’. The default is non–DOS mode. For backward compatibility, it is possible to use the option without the *mode* argument — then the default is used. Note that the optional *mode* argument cannot be separated from the **-c** option by a space, the correct form is for example **-c=dos**.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

-L, --color[=when]

Colorize the output. The optional argument *when* can be **auto**, **never** or **always**. If the *when* argument is omitted, it defaults to **auto**. The colors can be disabled; for the current built–in default see the **--help** output. See also the **COLORS** section.

-l, --list

List the partition tables for the specified devices and then exit.

If no devices are given, the devices mentioned in */proc/partitions* (if this file exists) are used. Devices are always listed in the order in which they are specified on the command-line, or by the kernel listed in */proc/partitions*.

-x, --list-details

Like **--list**, but provides more details.

--lock[=mode]

Use exclusive BSD lock for device or file it operates. The optional argument *mode* can be **yes**, **no** (or 1 and 0) or **nonblock**. If the *mode* argument is omitted, it defaults to **yes**. This option overwrites environment variable **\$LOCK_BLOCK_DEVICE**. The default is not to use any lock at all, but it's recommended to avoid collisions with **systemd-udevd(8)** or other tools.

-n, --noauto-pt

Don't automatically create a default partition table on empty device. The partition table has to be explicitly created by user (by command like 'o', 'g', etc.).

-o, --output list

Specify which output columns to print. Use **--help** to get a list of all supported columns.

The default list of columns may be extended if *list* is specified in the format *+list* (e.g., **-o +UUID**).

-s, --getsz

Print the size in 512-byte sectors of each given block device. This option is DEPRECATED in favour of **blockdev(8)**.

-t, --type type

Enable support only for disklabels of the specified *type*, and disable support for all other types.

-u, --units[=unit]

When listing partition tables, show sizes in 'sectors' or in 'cylinders'. The default is to show sizes in sectors. For backward compatibility, it is possible to use the option without the *unit* argument — then the default is used. Note that the optional *unit* argument cannot be separated from the **-u** option by a space, the correct form is for example '**-u=cylinders**'.

-C, --cylinders number

Specify the *number* of cylinders of the disk. I have no idea why anybody would want to do so.

-H, --heads number

Specify the number of heads of the disk. (Not the physical number, of course, but the number used for partition tables.) Reasonable values are 255 and 16.

-S, --sectors number

Specify the number of sectors per track of the disk. (Not the physical number, of course, but the number used for partition tables.) A reasonable value is 63.

-w, --wipe when

Wipe filesystem, RAID and partition-table signatures from the device, in order to avoid possible collisions. The argument *when* can be **auto**, **never** or **always**. When this option is not given, the default is **auto**, in which case signatures are wiped only when in interactive mode. In all cases detected signatures are reported by warning messages before a new partition table is created. See also **wipefs(8)**

command.

-W, --wipe-partitions *when*

Wipe filesystem, RAID and partition-table signatures from a newly created partitions, in order to avoid possible collisions. The argument *when* can be **auto**, **never** or **always**. When this option is not given, the default is **auto**, in which case signatures are wiped only when in interactive mode and after confirmation by user. In all cases detected signatures are reported by warning messages before a new partition is created. See also **wipefs(8)** command.

-V, --version

Display version information and exit.

DEVICES

The *device* is usually */dev/sda*, */dev/sdb* or so. A device name refers to the entire disk. Old systems without libata (a library used inside the Linux kernel to support ATA host controllers and devices) make a difference between IDE and SCSI disks. In such cases the device name will be */dev/hd** (IDE) or */dev/sd** (SCSI).

The *partition* is a device name followed by a partition number. For example, */dev/sda1* is the first partition on the first hard disk in the system. See also Linux kernel documentation (the *Documentation/admin-guide/devices.txt* file).

SIZES

The "last sector" dialog accepts partition size specified by number of sectors or by *+/-<size>{K,B,M,G,...}* notation.

If the size is prefixed by '+' then it is interpreted as relative to the partition first sector. If the size is prefixed by '-' then it is interpreted as relative to the high limit (last available sector for the partition).

In the case the size is specified in bytes than the number may be followed by the multiplicative suffixes KiB=1024, MiB=1024*1024, and so on for GiB, TiB, PiB, EiB, ZiB and YiB. The "iB" is optional, e.g., "K" has the same meaning as "KiB".

The relative sizes are always aligned according to device I/O limits. The *+/-<size>{K,B,M,G,...}* notation is recommended.

For backward compatibility **fdisk** also accepts the suffixes KB=1000, MB=1000*1000, and so on for GB, TB, PB, EB, ZB and YB. These 10^N suffixes are deprecated.

SCRIPT FILES

fdisk allows reading (by 'T' command) **sfdisk(8)** compatible script files. The script is applied to in-memory partition table, and then it is possible to modify the partition table before you write it to the device.

And vice-versa it is possible to write the current in-memory disk layout to the script file by command 'O'.

The script files are compatible between **cfdisk(8)**, **sfdisk(8)**, **fdisk** and other libfdisk applications. For more details see **sfdisk(8)**.

DISK LABELS

GPT (GUID Partition Table)

GPT is modern standard for the layout of the partition table. GPT uses 64-bit logical block addresses, checksums, UUIDs and names for partitions and an unlimited number of partitions (although the number of partitions is usually restricted to 128 in many partitioning tools).

Note that the first sector is still reserved for a **protective MBR** in the GPT specification. It prevents MBR-only partitioning tools from mis-recognizing and overwriting GPT disks.

GPT is always a better choice than MBR, especially on modern hardware with a UEFI boot loader.

DOS–type (MBR)

A DOS–type partition table can describe an unlimited number of partitions. In sector 0 there is room for the description of 4 partitions (called ‘primary’). One of these may be an extended partition; this is a box holding logical partitions, with descriptors found in a linked list of sectors, each preceding the corresponding logical partitions. The four primary partitions, present or not, get numbers 1–4. Logical partitions are numbered starting from 5.

In a DOS–type partition table the starting offset and the size of each partition is stored in two ways: as an absolute number of sectors (given in 32 bits), and as a **Cylinders/Heads/Sectors** triple (given in 10+8+6 bits). The former is OK — with 512–byte sectors this will work up to 2 TB. The latter has two problems. First, these C/H/S fields can be filled only when the number of heads and the number of sectors per track are known. And second, even if we know what these numbers should be, the 24 bits that are available do not suffice. DOS uses C/H/S only, Windows uses both, Linux never uses C/H/S. The **C/H/S addressing is deprecated** and may be unsupported in some later **fdisk** version.

Please, read the DOS–mode section if you want DOS–compatible partitions. **fdisk** does not care about cylinder boundaries by default.

BSD/Sun–type

A BSD/Sun disklabel can describe 8 partitions, the third of which should be a ‘whole disk’ partition. Do not start a partition that actually uses its first sector (like a swap partition) at cylinder 0, since that will destroy the disklabel. Note that a **BSD label** is usually nested within a DOS partition.

IRIX/SGI–type

An IRIX/SGI disklabel can describe 16 partitions, the eleventh of which should be an entire ‘volume’ partition, while the ninth should be labeled ‘volume header’. The volume header will also cover the partition table, i.e., it starts at block zero and extends by default over five cylinders. The remaining space in the volume header may be used by header directory entries. No partitions may overlap with the volume header. Also do not change its type or make some filesystem on it, since you will lose the partition table. Use this type of label only when working with Linux on IRIX/SGI machines or IRIX/SGI disks under Linux.

A **sync(2)** and an **ioctl(BLKRRPART)** (rereading the partition table from disk) are performed before exiting when the partition table has been updated.

DOS MODE AND DOS 6.X WARNING

Note that all this is deprecated. You don’t have to care about things like geometry and cylinders on modern operating systems. If you really want DOS–compatible partitioning then you have to enable DOS mode and cylinder units by using the ‘–c=dos –u=cylinders’ **fdisk command–line options.**

The DOS 6.x FORMAT command looks for some information in the first sector of the data area of the partition, and treats this information as more reliable than the information in the partition table. DOS FORMAT expects DOS FDISK to clear the first 512 bytes of the data area of a partition whenever a size change occurs. DOS FORMAT will look at this extra information even if the /U flag is given — we consider this a bug in DOS FORMAT and DOS FDISK.

The bottom line is that if you use **fdisk** or **cfdisk(8)** to change the size of a DOS partition table entry, then you must also use **dd(1)** to **zero the first 512 bytes** of that partition before using DOS FORMAT to format the partition. For example, if you were using **fdisk** to make a DOS partition table entry for **/dev/sda1**, then (after exiting **fdisk** and rebooting Linux so that the partition table information is valid) you would use the command **dd if=/dev/zero of=/dev/sda1 bs=512 count=1** to zero the first 512 bytes of the partition.

fdisk usually obtains the disk geometry automatically. This is not necessarily the physical disk geometry (indeed, modern disks do not really have anything like a physical geometry, certainly not something that can be described in the simplistic Cylinders/Heads/Sectors form), but it is the disk geometry that MS–DOS

uses for the partition table.

Usually all goes well by default, and there are no problems if Linux is the only system on the disk. However, if the disk has to be shared with other operating systems, it is often a good idea to let an **fdisk** from another operating system make at least one partition. When Linux boots it looks at the partition table, and tries to deduce what (fake) geometry is required for good cooperation with other systems.

Whenever a partition table is printed out in DOS mode, a consistency check is performed on the partition table entries. This check verifies that the physical and logical start and end points are identical, and that each partition starts and ends on a cylinder boundary (except for the first partition).

Some versions of MS-DOS create a first partition which does not begin on a cylinder boundary, but on sector 2 of the first cylinder. Partitions beginning in cylinder 1 cannot begin on a cylinder boundary, but this is unlikely to cause difficulty unless you have OS/2 on your machine.

For best results, you should always use an OS-specific partition table program. For example, you should make DOS partitions with the DOS FDISK program and Linux partitions with the Linux **fdisk** or Linux **cfdisk(8)** programs.

COLORS

The output colorization is implemented by **terminal-colors.d(5)** functionality. Implicit coloring can be disabled by an empty file

/etc/terminal-colors.d/fdisk.disable

for the **fdisk** command or for all tools by

/etc/terminal-colors.d/disable

The user-specific *\$XDG_CONFIG_HOME/terminal-colors.d* or *\$HOME/.config/terminal-colors.d* overrides the global setting.

Note that the output colorization may be enabled by default, and in this case *terminal-colors.d* directories do not have to exist yet.

The logical color names supported by **fdisk** are:

header

The header of the output tables.

help-title

The help section titles.

warn

The warning messages.

welcome

The welcome message.

ENVIRONMENT

FDISK_DEBUG=all

enables fdisk debug output.

LIBFDISK_DEBUG=all

enables libfdisk debug output.

LIBBLKID_DEBUG=all
enables libblkid debug output.

LIBSMARTCOLS_DEBUG=all
enables libsmartcols debug output.

LIBSMARTCOLS_DEBUG_PADDING=on
use visible padding characters.

LOCK_BLOCK_DEVICE=<mode>
use exclusive BSD lock. The mode is "1" or "0". See **--lock** for more details.

AUTHORS

Karel Zak <kzak@redhat.com>, Davidlohr Bueso <dave@gnu.org>

The original version was written by Andries E. Brouwer, A. V. Le Blanc and others.

SEE ALSO

[cfdisk\(8\)](#), [mkfs\(8\)](#), [partx\(8\)](#), [sfdisk\(8\)](#)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **fdisk** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

fg(1) - Linux man page

Name

bash, :, .., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg, getopt, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read, readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see **bash(1)**

Bash Builtin Commands

See Also

bash(1), **sh(1)**

NAME

find – search for files in a directory hierarchy

SYNOPSIS

find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

DESCRIPTION

This manual page documents the GNU version of **find**. GNU**find** searches the directory tree rooted at each given starting-point by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for *and* operations, true for *or*), at which point **find** moves on to the next file name. If no starting-point is specified, ‘.’ is assumed.

If you are using **find** in an environment where security is important (for example if you are using it to search directories that are writable by other users), you should read the ‘Security Considerations’ chapter of the findutils documentation, which is called **Finding Files** and comes with findutils. That document also includes a lot more detail and discussion than this manual page, so you may find it a more useful source of information.

OPTIONS

The **-H**, **-L** and **-P** options control the treatment of symbolic links. Command-line arguments following these are taken to be names of files or directories to be examined, up to the first argument that begins with ‘-’, or the argument ‘(’ or ‘!’. That argument and any following arguments are taken to be the expression describing what is to be searched for. If no paths are given, the current directory is used. If no expression is given, the expression **-print** is used (but you should probably consider using **-print0** instead, anyway).

This manual page talks about ‘options’ within the expression list. These options control the behaviour of **find** but are specified immediately after the last path name. The five ‘real’ options **-H**, **-L**, **-P**, **-D** and **-O** must appear before the first path name, if at all. A double dash -- could theoretically be used to signal that any remaining arguments are not options, but this does not really work due to the way **find** determines the end of the following path arguments: it does that by reading until an expression argument comes (which also starts with a ‘-’). Now, if a path argument would start with a ‘-’, then **find** would treat it as expression argument instead. Thus, to ensure that all start points are taken as such, and especially to prevent that wildcard patterns expanded by the calling shell are not mistakenly treated as expression arguments, it is generally safer to prefix wildcards or dubious path names with either ‘./’ or to use absolute path names starting with ‘/’.

- P** Never follow symbolic links. This is the default behaviour. When**find** examines or prints information about files, and the file is a symbolic link, the information used shall be taken from the properties of the symbolic link itself.
- L** Follow symbolic links. When **find** examines or prints information about files, the information used shall be taken from the properties of the file to which the link points, not from the link itself (unless it is a broken symbolic link or **find** is unable to examine the file to which the link points). Use of this option implies **-noleaf**. If you later use the **-P** option, **-noleaf** will still be in effect. If **-L** is in effect and **find** discovers a symbolic link to a subdirectory during its search, the subdirectory pointed to by the symbolic link will be searched.

When the **-L** option is in effect, the **-type** predicate will always match against the type of the file that a symbolic link points to rather than the link itself (unless the symbolic link is broken). Actions that can cause symbolic links to become broken while **find** is executing (for example **-delete**) can give rise to confusing behaviour. Using **-L** causes the **-lname** and **-ilname** predicates always to return false.
- H** Do not follow symbolic links, except while processing the command line arguments. When**find** examines or prints information about files, the information used shall be taken from the properties of the symbolic link itself. The only exception to this behaviour is when a file specified on the command line is a symbolic link, and the link can be resolved. For that situation, the information

used is taken from whatever the link points to (that is, the link is followed). The information about the link itself is used as a fallback if the file pointed to by the symbolic link cannot be examined. If **-H** is in effect and one of the paths specified on the command line is a symbolic link to a directory, the contents of that directory will be examined (though of course **-maxdepth 0** would prevent this).

If more than one of **-H**, **-L** and **-P** is specified, each overrides the others; the last one appearing on the command line takes effect. Since it is the default, the **-P** option should be considered to be in effect unless either **-H** or **-L** is specified.

GNU **find** frequently stats files during the processing of the command line itself, before any searching has begun. These options also affect how those arguments are processed. Specifically, there are a number of tests that compare files listed on the command line against a file we are currently considering. In each case, the file specified on the command line will have been examined and some of its properties will have been saved. If the named file is in fact a symbolic link, and the **-P** option is in effect (or if neither **-H** nor **-L** were specified), the information used for the comparison will be taken from the properties of the symbolic link. Otherwise, it will be taken from the properties of the file the link points to. If **find** cannot follow the link (for example because it has insufficient privileges or the link points to a nonexistent file) the properties of the link itself will be used.

When the **-H** or **-L** options are in effect, any symbolic links listed as the argument of **-newer** will be dereferenced, and the timestamp will be taken from the file to which the symbolic link points. The same consideration applies to **-newerXY**, **-anewer** and **-cnewer**.

The **-follow** option has a similar effect to **-L**, though it takes effect at the point where it appears (that is, if **-L** is not used but **-follow** is, any symbolic links appearing after **-follow** on the command line will be dereferenced, and those before it will not).

-D debugopts

Print diagnostic information; this can be helpful to diagnose problems with why **find** is not doing what you want. The list of debug options should be comma separated. Compatibility of the debug options is not guaranteed between releases of findutils. For a complete list of valid debug options, see the output of **find -D help**. Valid debug options include

- | | |
|--------|--|
| exec | Show diagnostic information relating to -exec , -execdir , -ok and -okdir |
| opt | Prints diagnostic information relating to the optimisation of the expression tree; see the -O option. |
| rates | Prints a summary indicating how often each predicate succeeded or failed. |
| search | Navigate the directory tree verbosely. |
| stat | Print messages as files are examined with the stat and Istat system calls. The find program tries to minimise such calls. |
| tree | Show the expression tree in its original and optimised form. |
| all | Enable all of the other debug options (but help). |
| help | Explain the debugging options. |

-Olevel

Enables query optimisation. The **find** program reorders tests to speed up execution while preserving the overall effect; that is, predicates with side effects are not reordered relative to each other. The optimisations performed at each optimisation level are as follows.

- | | |
|---|--|
| 0 | Equivalent to optimisation level 1. |
| 1 | This is the default optimisation level and corresponds to the traditional behaviour. Expressions are reordered so that tests based only on the names of files (for example -name and -regex) are performed first. |

- 2 Any **-type** or **-xtype** tests are performed after any tests based only on the names of files, but before any tests that require information from the inode. On many modern versions of Unix, file types are returned by **readdir()** and so these predicates are faster to evaluate than predicates which need to stat the file first. If you use the **-fstype** *FOO* predicate and specify a filesystem type *FOO* which is not known (that is, present in '/etc/mtab') at the time **find** starts, that predicate is equivalent to **-false**.
- 3 At this optimisation level, the full cost-based query optimiser is enabled. The order of tests is modified so that cheap (i.e. fast) tests are performed first and more expensive ones are performed later, if necessary. Within each cost band, predicates are evaluated earlier or later according to whether they are likely to succeed or not. For **-o**, predicates which are likely to succeed are evaluated earlier, and for **-a**, predicates which are likely to fail are evaluated earlier.

The cost-based optimiser has a fixed idea of how likely any given test is to succeed. In some cases the probability takes account of the specific nature of the test (for example, **-type f** is assumed to be more likely to succeed than **-type c**). The cost-based optimiser is currently being evaluated. If it does not actually improve the performance of **find**, it will be removed again. Conversely, optimisations that prove to be reliable, robust and effective may be enabled at lower optimisation levels over time. However, the default behaviour (i.e. optimisation level 1) will not be changed in the 4.3.x release series. The findutils test suite runs all the tests on **find** at each optimisation level and ensures that the result is the same.

EXPRESSION

The part of the command line after the list of starting points is the *expression*. This is a kind of query specification describing how we match files and what we do with the files that were matched. An expression is composed of a sequence of things:

Tests Tests return a true or false value, usually on the basis of some property of a file we are considering. The **-empty** test for example is true only when the current file is empty.

Actions

Actions have side effects (such as printing something on the standard output) and return either true or false, usually based on whether or not they are successful. The **-print** action for example prints the name of the current file on the standard output.

Global options

Global options affect the operation of tests and actions specified on any part of the command line. Global options always return true. The **-depth** option for example makes **find** traverse the file system in a depth-first order.

Positional options

Positional options affect only tests or actions which follow them. Positional options always return true. The **-r egextype** option for example is positional, specifying the regular expression dialect for regular expressions occurring later on the command line.

Operators

Operators join together the other items within the expression. They include for example **-o** (meaning logical OR) and **-a** (meaning logical AND). Where an operator is missing, **-a** is assumed.

The **-print** action is performed on all files for which the whole expression is true, unless it contains an action other than **-prune** or **-quit**. Actions which inhibit the default **-print** are **-delete**, **-exec**, **-execdir**, **-ok**, **-okdir**, **-fls**, **-fprintf**, **-fprintf**, **-ls**, **-print** and **-printf**.

The **-delete** action also acts like an option (since it implies **-depth**).

POSITIONAL OPTIONS

Positional options always return true. They affect only tests occurring later on the command line.

-daystart

Measure times (for **-amin**, **-atime**, **-cmin**, **-ctime**, **-mmin**, and **-mtime**) from the beginning of today rather than from 24 hours ago. This option only affects tests which appear later on the command line.

-follow

Deprecated; use the **-L** option instead. Dereference symbolic links. Implies **-oleaf**. The **-f follow** option affects only those tests which appear after it on the command line. Unless the **-H** or **-L** option has been specified, the position of the **-follow** option changes the behaviour of the **-newer** predicate; any files listed as the argument of **-newer** will be dereferenced if they are symbolic links. The same consideration applies to **-newerXY**, **-anewer** and **-cnewer**. Similarly, the **-type** predicate will always match against the type of the file that a symbolic link points to rather than the link itself. Using **-follow** causes the **-lname** and **-ilname** predicates always to return false.

-regextype type

Changes the regular expression syntax understood by **-regex** and **-iregex** tests which occur later on the command line. To see which regular expression types are known, use **-regextype help**. The Texinfo documentation (see **SEE ALSO**) explains the meaning of and differences between the various types of regular expression.

-warn, -nowarn

Turn warning messages on or off. These warnings apply only to the command line usage, not to any conditions that **find** might encounter when it searches directories. The default behaviour corresponds to **-warn** if standard input is a tty, and to **-nowarn** otherwise. If a warning message relating to command-line usage is produced, the exit status of **find** is not affected. If the **POSIXLY_CORRECT** environment variable is set, and **-warn** is also used, it is not specified which, if any, warnings will be active.

GLOBAL OPTIONS

Global options always return true. Global options take effect even for tests which occur earlier on the command line. To prevent confusion, global options should be specified on the command-line after the list of start points, just before the first test, positional option or action. If you specify a global option in some other place, **find** will issue a warning message explaining that this can be confusing.

The global options occur after the list of start points, and so are not the same kind of option as **-L**, for example.

-d A synonym for **-depth**, for compatibility with FreeBSD, NetBSD, MacOS X and OpenBSD.

-depth Process each directory's contents before the directory itself. The **-delete** action also implies **-depth**.

-help, --help

Print a summary of the command-line usage of **find** and exit.

-ignore_readdir_race

Normally, **find** will emit an error message when it fails to stat a file. If you give this option and a file is deleted between the time **find** reads the name of the file from the directory and the time it tries to stat the file, no error message will be issued. This also applies to files or directories whose names are given on the command line. This option takes effect at the time the command line is read, which means that you cannot search one part of the filesystem with this option on and part of it with this option off (if you need to do that, you will need to issue two **find** commands instead, one with the option and one without it).

Furthermore, **find** with the **-ignore_readdir_race** option will ignore errors of the **-delete** action in the case the file has disappeared since the parent directory was read: it will not output an error diagnostic, and the return code of the **-delete** action will be true.

-maxdepth levels

Descend at most *levels* (a non-negative integer) levels of directories below the starting-points. Using **-maxdepth 0** means only apply the tests and actions to the starting-points themselves.

-mindepth levels

Do not apply any tests or actions at levels less than *levels* (a non-negative integer). Using **-mindepth 1** means process all files except the starting-points.

-mount

Don't descend directories on other filesystems. An alternate name for **-xdev**, for compatibility with some other versions of **find**.

-noignore_readdir_race

Turns off the effect of **-ignore_readdir_race**.

-noleaf Do not optimize by assuming that directories contain 2 fewer subdirectories than their hard link count. This option is needed when searching filesystems that do not follow the Unix directory-link convention, such as CD-ROM or MS-DOS filesystems or AFS volume mount points. Each directory on a normal Unix filesystem has at least 2 hard links: its name and its '.' entry. Additionally, its subdirectories (if any) each have a '..' entry linked to that directory. When **find** is examining a directory, after it has statted 2 fewer subdirectories than the directory's link count, it knows that the rest of the entries in the directory are non-directories ('leaf' files in the directory tree). If only the files' names need to be examined, there is no need to stat them; this gives a significant increase in search speed.

-version, --version

Print the **find** version number and exit.

-xdev Don't descend directories on other filesystems.

TESTS

Some tests, for example **-newerXY** and **-samefile**, allow comparison between the file currently being examined and some reference file specified on the command line. When these tests are used, the interpretation of the reference file is determined by the options **-H**, **-L** and **-P** and any previous **-follow**, but the reference file is only examined once, at the time the command line is parsed. If the reference file cannot be examined (for example, the **stat(2)** system call fails for it), an error message is issued, and **find** exits with a nonzero status.

A numeric argument *n* can be specified to tests (like **-amin**, **-mtime**, **-gid**, **-inum**, **-links**, **-size**, **-uid** and

-used) as

- +n for greater than *n*,
- n for less than *n*,
- n* for exactly *n*.

Supported tests:

-amin *n*

File was last accessed less than, more than or exactly *n* minutes ago.

-anewer *reference*

Time of the last access of the current file is more recent than that of the last data modification of the *reference* file. If *reference* is a symbolic link and the **-H** option or the **-L** option is in effect, then the time of the last data modification of the file it points to is always used.

-atime *n*

File was last accessed less than, more than or exactly *n**24 hours ago. When find figures out how many 24-hour periods ago the file was last accessed, any fractional part is ignored, so to match **-atime +1**, a file has to have been accessed at least *two* days ago.

-cmin *n*

File's status was last changed less than, more than or exactly *n* minutes ago.

-cnewer *reference*

Time of the last status change of the current file is more recent than that of the last data modification of the *reference* file. If *reference* is a symbolic link and the **-H** option or the **-L** option is in effect, then the time of the last data modification of the file it points to is always used.

-ctime *n*

File's status was last changed less than, more than or exactly *n**24 hours ago. See the comments for **-atime** to understand how rounding affects the interpretation of file status change times.

-empty File is empty and is either a regular file or a directory.

-executable

Matches files which are executable and directories which are searchable (in a file name resolution sense) by the current user. This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access(2)** system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access(2)** in the client's kernel and so cannot make use of the UID mapping information held on the server. Because this test is based only on the result of the **access(2)** system call, there is no guarantee that a file for which this test succeeds can actually be executed.

-false Always false.

-fstype *type*

File is on a filesystem of type *type*. The valid filesystem types vary among different versions of Unix; an incomplete list of filesystem types that are accepted on some version of Unix or another is: ufs, 4.2, 4.3, nfs, tmp, mfs, S51K, S52K. You can use **-printf** with the %F directive to see the types of your filesystems.

-gid *n* File's numeric group ID is less than, more than or exactly *n*.

-group *gname*

File belongs to group *gname* (numeric group ID allowed).

-ilname *pattern*

Like **-lname**, but the match is case insensitive. If the **-L** option or the **-f follow** option is in effect, this test returns false unless the symbolic link is broken.

-iname *pattern*

Like **-name**, but the match is case insensitive. For example, the patterns ‘fo*’ and ‘F??’ match the file names ‘Foo’, ‘FOO’, ‘foo’, ‘fOo’, etc. The pattern ‘*foo*’ will also match a file called ‘.foobar’.

-inum *n*

File has inode number smaller than, greater than or exactly *n*. It is normally easier to use the **-samefile** test instead.

-ipath *pattern*

Like **-path**, but the match is case insensitive.

-iregex *pattern*

Like **-regex**, but the match is case insensitive.

-iwholename *pattern*

See **-ipath**. This alternative is less portable than **-ipath**.

-links *n*

File has less than, more than or exactly *n* hard links.

-lname *pattern*

File is a symbolic link whose contents match shell pattern *pattern*. The metacharacters do not treat ‘/’ or ‘.’ specially. If the **-L** option or the **-f follow** option is in effect, this test returns false unless the symbolic link is broken.

-mmin *n*

File's data was last modified less than, more than or exactly *n* minutes ago.

-mtime *n*

File's data was last modified less than, more than or exactly *n**24 hours ago. See the comments for **-atime** to understand how rounding affects the interpretation of file modification times.

-name *pattern*

Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. Because the leading directories are removed, the file names considered for a match with **-name** will never include a slash, so ‘-name a/b’ will never match anything (you probably need to use **-path** instead). A warning is issued if you try to do this, unless the environment variable **POSIXLY_CORRECT** is set. The metacharacters (‘*’, ‘?’, and ‘[]’) match a ‘.’ at the start of the base name (this is a change in findutils-4.2.2; see section STANDARDS CONFORMANCE

below). To ignore a directory and the files under it, use **-prune** rather than checking every file in the tree; see an example in the description of that action. Braces are not recognised as being special, despite the fact that some shells including Bash imbue braces with a special meaning in shell patterns. The filename matching is performed with the use of the **fnmatch(3)** library function. Don't forget to enclose the pattern in quotes in order to protect it from expansion by the shell.

-newer *reference*

Time of the last data modification of the current file is more recent than that of the last data modification of the *reference* file. If *reference* is a symbolic link and the **-H** option or the **-L** option is in effect, then the time of the last data modification of the file it points to is always used.

-newerXY *reference*

Succeeds if timestamp *X* of the file being considered is newer than timestamp *Y* of the file *reference*. The letters *X* and *Y* can be any of the following letters:

- a The access time of the file *reference*
- B The birth time of the file *reference*
- c The inode status change time of *reference*
- m The modification time of the file *reference*
- t *reference* is interpreted directly as a time

Some combinations are invalid; for example, it is invalid for *X* to be *t*. Some combinations are not implemented on all systems; for example *B* is not supported on all systems. If an invalid or unsupported combination of *XY* is specified, a fatal error results. Time specifications are interpreted as for the argument to the **-d** option of GNU **date**. If you try to use the birth time of a reference file, and the birth time cannot be determined, a fatal error message results. If you specify a test which refers to the birth time of files being examined, this test will fail for any files where the birth time is unknown.

-nogroup

No group corresponds to file's numeric group ID.

-nouser

No user corresponds to file's numeric user ID.

-path *pattern*

File name matches shell pattern *pattern*. The metacharacters do not treat '/' or '.' specially; so, for example,

```
find . -path "./sr*sc"
```

will print an entry for a directory called *./src/misc* (if one exists). To ignore a whole directory tree, use **-prune** rather than checking every file in the tree. Note that the pattern match test applies to the whole file name, starting from one of the start points named on the command line. It would only make sense to use an absolute path name here if the relevant start point is also an absolute path. This means that this command will never match anything:

```
find bar -path /foo/bar/myfile -print
```

Find compares the **-path** argument with the concatenation of a directory name and the base name of the file it's examining. Since the concatenation will never end with a slash, **-path** arguments ending in a slash will match nothing (except perhaps a start point specified on the command line). The predicate **-path** is also supported by HP-UX **find** and is part of the POSIX 2008 standard.

-perm mode

File's permission bits are exactly *mode* (octal or symbolic). Since an exact match is required, if you want to use this form for symbolic modes, you may have to specify a rather complex mode string. For example ‘-perm g=w’ will only match files which have mode 0020 (that is, ones for which group write permission is the only permission set). It is more likely that you will want to use the ‘/’ or ‘-’ forms, for example ‘-perm -g=w’, which matches any file with group write permission. See the **EXAMPLES** section for some illustrative examples.

-perm -mode

All of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form, and this is usually the way in which you would want to use them. You must specify ‘u’, ‘g’ or ‘o’ if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples.

-perm /mode

Any of the permission bits *mode* are set for the file. Symbolic modes are accepted in this form. You must specify ‘u’, ‘g’ or ‘o’ if you use a symbolic mode. See the **EXAMPLES** section for some illustrative examples. If no permission bits in *mode* are set, this test matches any file (the idea here is to be consistent with the behaviour of **-perm -000**).

-perm +mode

This is no longer supported (and has been deprecated since 2005). Use **-perm /mode** instead.

-readable

Matches files which are readable by the current user. This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access(2)** system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access(2)** in the client's kernel and so cannot make use of the UID mapping information held on the server.

-regex pattern

File name matches regular expression *pattern*. This is a match on the whole path, not a search. For example, to match a file named */fubar3*, you can use the regular expression ‘.*bar.’ or ‘.*b.*3’, but not ‘f.*r3’. The regular expressions understood by **find** are by default Emacs Regular Expressions (except that ‘.’ matches newline), but this can be changed with the **-regextype** option.

-samefile name

File refers to the same inode as *name*. When **-L** is in effect, this can include symbolic links.

-size n[cwbkMG]

File uses less than, more than or exactly *n* units of space, rounding up. The following suffixes can be used:

- ‘b’ for 512-byte blocks (this is the default if no suffix is used)
- ‘c’ for bytes
- ‘w’ for two-byte words
- ‘k’ for kibibytes (KiB, units of 1024 bytes)
- ‘M’ for mebibytes (MiB, units of 1024 * 1024 = 1 048 576 bytes)
- ‘G’ for gibibytes (GiB, units of 1024 * 1024 * 1024 = 1 073 741 824 bytes)

The size is simply the *st_size* member of the struct *stat* populated by the *Istat* (or *stat*) system call, rounded up as shown above. In other words, it's consistent with the result you get for **ls -l**. Bear

in mind that the ‘%k’ and ‘%b’ format specifiers of **-printf** handle sparse files differently. The ‘b’ suffix always denotes 512-byte blocks and never 1024-byte blocks, which is different to the behaviour of **-ls**.

The + and - prefixes signify greater than and less than, as usual; i.e., an exact size of *n* units does not match. Bear in mind that the size is rounded up to the next unit. Therefore **-size -1M** is not equivalent to **-size -1048576c**. The former only matches empty files, the latter matches files from 0 to 1,048,575 bytes.

-true Always true.

-type *c* File is of type *c*:

b	block (buffered) special
c	character (unbuffered) special
d	directory
p	named pipe (FIFO)
f	regular file
l	symbolic link; this is never true if the -L option or the -follow option is in effect, unless the symbolic link is broken. If you want to search for symbolic links when -L is in effect, use -xtype .
s	socket
D	door (Solaris)

To search for more than one type at once, you can supply the combined list of type letters separated by a comma ‘,’ (GNU extension).

-uid *n* File’s numeric user ID is less than, more than or exactly *n*.

-used *n*

File was last accessed less than, more than or exactly *n* days after its status was last changed.

-user *uname*

File is owned by user *uname* (numeric user ID allowed).

-wholename *pattern*

See **-path**. This alternative is less portable than **-path**.

-writable

Matches files which are writable by the current user. This takes into account access control lists and other permissions artefacts which the **-perm** test ignores. This test makes use of the **access(2)** system call, and so can be fooled by NFS servers which do UID mapping (or root-squashing), since many systems implement **access(2)** in the client’s kernel and so cannot make use of the UID mapping information held on the server.

-xtype *c*

The same as **-type** unless the file is a symbolic link. For symbolic links: if the **-H** or **-P** option was specified, true if the file is a link to a file of type *c*; if the **-L** option has been given, true if *c* is ‘l’. In other words, for symbolic links, **-xtype** checks the type of the file that **-type** does not check.

-context *pattern*

(SELinux only) Security context of the file matches glob *pattern*.

ACTIONS

- delete** Delete files; true if removal succeeded. If the removal failed, an error message is issued. If **-delete** fails, **find**'s exit status will be nonzero (when it eventually exits). Use of **-delete** automatically turns on the '**-depth**' option.

Warnings: Don't forget that the **find** command line is evaluated as an expression, so putting **-delete** first will make **find** try to delete everything below the starting points you specified. When testing a **find** command line that you later intend to use with **-delete**, you should explicitly specify **-depth** in order to avoid later surprises. Because **-delete** implies **-depth**, you cannot usefully use **-prune** and **-delete** together.

Together with the **-ignore_readdir_race** option, **find** will ignore errors of the **-delete** action in the case the file has disappeared since the parent directory was read: it will not output an error diagnostic, and the return code of the **-delete** action will be true.

-exec *command* ;

Execute *command*; true if 0 status is returned. All following arguments to **find** are taken to be arguments to the command until an argument consisting of ';' is encountered. The string '{ }' is replaced by the current file name being processed everywhere it occurs in the arguments to the command, not just in arguments where it is alone, as in some versions of **find**. Both of these constructions might need to be escaped (with a '\') or quoted to protect them from expansion by the shell. See the **EXAMPLES** section for examples of the use of the **-exec** option. The specified command is run once for each matched file. The command is executed in the starting directory. There are unavoidable security problems surrounding use of the **-exec** action; you should use the **-execdir** option instead.

-exec *command* { } +

This variant of the **-exec** action runs the specified command on the selected files, but the command line is built by appending each selected file name at the end; the total number of invocations of the command will be much less than the number of matched files. The command line is built in much the same way that **xargs** builds its command lines. Only one instance of '{ }' is allowed within the command, and it must appear at the end, immediately before the '+'; it needs to be escaped (with a '\') or quoted to protect it from interpretation by the shell. The command is executed in the starting directory. If any invocation with the '+' form returns a non-zero value as exit status, then **find** returns a non-zero exit status. If **find** encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. For this reason **-exec my-command ... { } + -quit** may not result in *my-command* actually being run. This variant of **-exec** always returns true.

-execdir *command* ;**-execdir *command* { } +**

Like **-exec**, but the specified command is run from the subdirectory containing the matched file, which is not normally the directory in which you started **find**. As with **-exec**, the '{ }' should be quoted if **find** is being invoked from a shell. This a much more secure method for invoking commands, as it avoids race conditions during resolution of the paths to the matched files. As with the **-exec** action, the '+' form of **-execdir** will build a command line to process more than one matched file, but any given invocation of *command* will only list files that exist in the same subdirectory. If you use this option, you must ensure that your \$PATH environment variable does not reference ':'; otherwise, an attacker can run any commands they like by leaving an appropriately-

named file in a directory in which you will run **-execdir**. The same applies to having entries in **\$PATH** which are empty or which are not absolute directory names. If any invocation with the ‘+’ form returns a non-zero value as exit status, then **find** returns a non-zero exit status. If **find** encounters an error, this can sometimes cause an immediate exit, so some pending commands may not be run at all. The result of the action depends on whether the + or the ; variant is being used; **-execdir command {} +** always returns true, while **-execdir command {} ;** returns true only if *command* returns 0.

-fls *file* True; like **-ls** but write to *file* like **-fprintf**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-fprintf *file*

True; print the full file name into file *file*. If *file* does not exist when **find** is run, it is created; if it does exist, it is truncated. The file names */dev/stdout* and */dev/stderr* are handled specially; they refer to the standard output and standard error output, respectively. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-fprintf0 *file*

True; like **-print0** but write to *file* like **-fprintf**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-printf *file format*

True; like **-printf** but write to *file* like **-fprintf**. The output file is always created, even if the predicate is never matched. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-ls

True; list current file in **ls -dils** format on standard output. The block counts are of 1 KB blocks, unless the environment variable **POSIXLY_CORRECT** is set, in which case 512-byte blocks are used. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-ok *command* ;

Like **-exec** but ask the user first. If the user agrees, run the command. Otherwise just return false. If the command is run, its standard input is redirected from */dev/null*.

The response to the prompt is matched against a pair of regular expressions to determine if it is an affirmative or negative response. This regular expression is obtained from the system if the ‘**POSIXLY_CORRECT**’ environment variable is set, or otherwise from **find**’s message translations. If the system has no suitable definition, **find**’s own definition will be used. In either case, the interpretation of the regular expression itself will be affected by the environment variables ‘**LC_CTYPE**’ (character classes) and ‘**LC_COLLATE**’ (character ranges and equivalence classes).

-okdir *command* ;

Like **-execdir** but ask the user first in the same way as for **-ok**. If the user does not agree, just return false. If the command is run, its standard input is redirected from */dev/null*.

-print True; print the full file name on the standard output, followed by a newline. If you are piping the output of **find** into another program and there is the faintest possibility that the files which you are searching for might contain a newline, then you should seriously consider using the **-print0** option instead of **-print**. See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

-print0 True; print the full file name on the standard output, followed by a null character (instead of the newline character that **-print** uses). This allows file names that contain newlines or other types of white space to be correctly interpreted by programs that process the **find** output. This option corresponds to the **-0** option of **xargs**.

-printf *format*

True; print *format* on the standard output, interpreting ‘\’ escapes and ‘%’ directives. Field widths and precisions can be specified as with the **printf(3)** C function. Please note that many of the fields are printed as %s rather than %d, and this may mean that flags don’t work as you might expect. This also means that the ‘-’ flag does work (it forces fields to be left-aligned). Unlike **-print**, **-printf** does not add a newline at the end of the string. The escapes and directives are:

\a	Alarm bell.
\b	Backspace.
\c	Stop printing from this format immediately and flush the output.
\f	Form feed.
\n	Newline.
\r	Carriage return.
\t	Horizontal tab.
\v	Vertical tab.
\0	ASCII NUL.
\\\	A literal backslash (‘\’).

\NNN The character whose ASCII code is NNN (octal).

A ‘\’ character followed by any other character is treated as an ordinary character, so they both are printed.

%% A literal percent sign.

%a File’s last access time in the format returned by the C **ctime(3)** function.

%Ak File’s last access time in the format specified by *k*, which is either ‘@’ or a directive for the C **strftime(3)** function. The following shows an incomplete list of possible values for *k*. Please refer to the documentation of **strftime(3)** for the full list. Some of the conversion specification characters might not be available on all systems, due to differences in the implementation of the **strftime(3)** library function.

@ seconds since Jan. 1, 1970, 00:00 GMT, with fractional part.

Time fields:

H	hour (00..23)
I	hour (01..12)
k	hour (0..23)
l	hour (1..12)
M	minute (00..59)

p	locale's AM or PM
r	time, 12-hour (hh:mm:ss [AP]M)
S	Second (00.00 .. 61.00). There is a fractional part.
T	time, 24-hour (hh:mm:ss.xxxxxxxxxx)
+	Date and time, separated by '+', for example '2004-04-28+22:22:05.0'. This is a GNU extension. The time is given in the current timezone (which may be affected by setting the TZ environment variable). The seconds field includes a fractional part.
X	locale's time representation (H:M:S). The seconds field includes a fractional part.
Z	time zone (e.g., EDT), or nothing if no time zone is determinable
Date fields:	
a	locale's abbreviated weekday name (Sun..Sat)
A	locale's full weekday name, variable length (Sunday..Saturday)
b	locale's abbreviated month name (Jan..Dec)
B	locale's full month name, variable length (January..December)
c	locale's date and time (Sat Nov 04 12:02:33 EST 1989). The format is the same as for ctime(3) and so to preserve compatibility with that format, there is no fractional part in the seconds field.
d	day of month (01..31)
D	date (mm/dd/yy)
F	date (yyyy-mm-dd)
h	same as b
j	day of year (001..366)
m	month (01..12)
U	week number of year with Sunday as first day of week (00..53)
w	day of week (0..6)
W	week number of year with Monday as first day of week (00..53)
x	locale's date representation (mm/dd/yy)
y	last two digits of year (00..99)
Y	year (1970...)
%b	The amount of disk space used for this file in 512-byte blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/512, but it can also be smaller if the file is a sparse file.
%c	File's last status change time in the format returned by the C ctime(3) function.
%Ck	File's last status change time in the format specified by k, which is the same as for %A.
%d	File's depth in the directory tree; 0 means the file is a starting-point.
%D	The device number on which the file exists (the st_dev field of struct stat), in decimal.
%f	Print the basename; the file's name with any leading directories removed (only the last element). For /, the result is '/'. See the EXAMPLES section for an example.

%F	Type of the filesystem the file is on; this value can be used for <code>-fstype</code> .
%g	File's group name, or numeric group ID if the group has no name.
%G	File's numeric group ID.
%h	Dirname; the Leading directories of the file's name (all but the last element). If the file name contains no slashes (since it is in the current directory) the %h specifier expands to '..'. For files which are themselves directories and contain a slash (including <code>/</code>), %h expands to the empty string. See the EXAMPLES section for an example.
%H	Starting-point under which file was found.
%i	File's inode number (in decimal).
%k	The amount of disk space used for this file in 1 KB blocks. Since disk space is allocated in multiples of the filesystem block size this is usually greater than %s/1024, but it can also be smaller if the file is a sparse file.
%l	Object of symbolic link (empty string if file is not a symbolic link).
%m	File's permission bits (in octal). This option uses the 'traditional' numbers which most Unix implementations use, but if your particular implementation uses an unusual ordering of octal permissions bits, you will see a difference between the actual value of the file's mode and the output of %m. Normally you will want to have a leading zero on this number, and to do this, you should use the # flag (as in, for example, '%#m').
%M	File's permissions (in symbolic form, as for <code>ls</code>). This directive is supported in findutils 4.2.5 and later.
%n	Number of hard links to file.
%p	File's name.
%P	File's name with the name of the starting-point under which it was found removed.
%s	File's size in bytes.
%S	File's sparseness. This is calculated as (BLOCKSIZE*st_blocks / st_size). The exact value you will get for an ordinary file of a certain length is system-dependent. However, normally sparse files will have values less than 1.0, and files which use indirect blocks may have a value which is greater than 1.0. In general the number of blocks used by a file is file system dependent. The value used for BLOCKSIZE is system-dependent, but is usually 512 bytes. If the file size is zero, the value printed is undefined. On systems which lack support for st_blocks, a file's sparseness is assumed to be 1.0.
%t	File's last modification time in the format returned by the C <code>ctime(3)</code> function.
%Tk	File's last modification time in the format specified by <i>k</i> , which is the same as for %A.
%u	File's user name, or numeric user ID if the user has no name.
%U	File's numeric user ID.
%y	File's type (like in <code>ls -l</code>), U=unknown type (shouldn't happen)
%Y	File's type (like %y), plus follow symbolic links: 'L'=loop, 'N'=nonexistent, '?' for any other error when determining the type of the target of a symbolic link.
%Z	(SELinux only) file's security context.
%{ %[%(%	Reserved for future use.

A '%' character followed by any other character is discarded, but the other character is printed (don't rely on this, as further format characters may be introduced). A '%' at the end of the format argument causes undefined behaviour since there is no following character. In some locales, it may hide your door keys, while in others it may remove the final page from the novel you are

reading.

The %m and %d directives support the #, 0 and + flags, but the other directives do not, even if they print numbers. Numeric directives that do not support these flags include **G**, **U**, **b**, **D**, **k** and **n**. The ‘-’ format flag is supported and changes the alignment of a field from right-justified (which is the default) to left-justified.

See the **UNUSUAL FILENAMES** section for information about how unusual characters in filenames are handled.

- prune True; if the file is a directory, do not descend into it. If **-depth** is given, then **-prune** has no effect. Because **-delete** implies **-depth**, you cannot usefully use **-prune** and **-delete** together. For example, to skip the directory *src/emacs* and all files and directories under it, and print the names of the other files found, do something like this:

```
find . -path ./src/emacs -prune -o -print
```

- quit Exit immediately (with return value zero if no errors have occurred). This is different to **-prune** because **-prune** only applies to the contents of pruned directories, while **-quit** simply makes **find** stop immediately. No child processes will be left running. Any command lines which have been built by **-exec ... +** or **-execdir ... +** are invoked before the program is exited. After **-quit** is executed, no more files specified on the command line will be processed. For example, ‘**find /tmp/foo /tmp/bar -print -quit**’ will print only ‘/tmp/foo’.

One common use of **-quit** is to stop searching the file system once we have found what we want. For example, if we want to find just a single file we can do this:

```
find / -name needle -print -quit
```

OPERATORS

Listed in order of decreasing precedence:

(*expr*) Force precedence. Since parentheses are special to the shell, you will normally need to quote them. Many of the examples in this manual page use backslashes for this purpose: ‘\(...\)' instead of ‘(...)'.

! *expr* True if *expr* is false. This character will also usually need protection from interpretation by the shell.

-not *expr*

Same as ! *expr*, but not POSIX compliant.

expr1 expr2

Two expressions in a row are taken to be joined with an implied **-a**; *expr2* is not evaluated if *expr1* is false.

expr1 -a expr2

Same as *expr1 expr2*.

expr1 -and expr2

Same as *expr1 expr2*, but not POSIX compliant.

expr1 –o *expr2*

Or; *expr2* is not evaluated if *expr1* is true.

expr1 –or *expr2*

Same as *expr1* –o *expr2*, but not POSIX compliant.

expr1 , *expr2*

List; both *expr1* and *expr2* are always evaluated. The value of *expr1* is discarded; the value of the list is the value of *expr2*. The comma operator can be useful for searching for several different types of thing, but traversing the filesystem hierarchy only once. The –fprintf action can be used to list the various matched items into several different output files.

Please note that –a when specified implicitly (for example by two tests appearing without an explicit operator between them) or explicitly has higher precedence than –o. This means that **find . -name afile -o -name bfile -print** will never print *afile*.

UNUSUAL FILENAMES

Many of the actions of **find** result in the printing of data which is under the control of other users. This includes file names, sizes, modification times and so forth. File names are a potential problem since they can contain any character except ‘\0’ and ‘/’. Unusual characters in file names can do unexpected and often undesirable things to your terminal (for example, changing the settings of your function keys on some terminals). Unusual characters are handled differently by various actions, as described below.

–print0, –fprint0

Always print the exact filename, unchanged, even if the output is going to a terminal.

–ls, –fls

Unusual characters are always escaped. White space, backslash, and double quote characters are printed using C-style escaping (for example ‘f’, ‘\”’). Other unusual characters are printed using an octal escape. Other printable characters (for –ls and –fls these are the characters between octal 041 and 0176) are printed as-is.

–printf, –fprintf

If the output is not going to a terminal, it is printed as-is. Otherwise, the result depends on which directive is in use. The directives %D, %F, %g, %G, %H, %Y, and %y expand to values which are not under control of files’ owners, and so are printed as-is. The directives %a, %b, %c, %d, %i, %k, %m, %M, %n, %s, %t, %u and %U have values which are under the control of files’ owners but which cannot be used to send arbitrary data to the terminal, and so these are printed as-is. The directives %f, %h, %l, %p and %P are quoted. This quoting is performed in the same way as for GNU ls. This is not the same quoting mechanism as the one used for –ls and –fls. If you are able to decide what format to use for the output of **find** then it is normally better to use ‘\0’ as a terminator than to use newline, as file names can contain white space and newline characters. The setting of the ‘LC_CTYPE’ environment variable is used to determine which characters need to be quoted.

–print, –fprint

Quoting is handled in the same way as for –printf and –fprintf. If you are using **find** in a script or in a situation where the matched files might have arbitrary names, you should consider using –print0 instead of –print.

The –ok and –okdir actions print the current filename as-is. This may change in a future release.

STANDARDS CONFORMANCE

For closest compliance to the POSIX standard, you should set the `POSIXLY_CORRECT` environment variable. The following options are specified in the POSIX standard (IEEE Std 1003.1-2008, 2016 Edition):

-H This option is supported.

-L This option is supported.

-name This option is supported, but POSIX conformance depends on the POSIX conformance of the system's `fnmatch(3)` library function. As of `findutils`-4.2.2, shell metacharacters ('*', '?' or '[]' for example) match a leading '.', because IEEE PASC interpretation 126 requires this. This is a change from previous versions of `findutils`.

-type Supported. POSIX specifies 'b', 'c', 'd', 'l', 'p', 'f' and 's'. GNU `find` also supports 'D', representing a Door, where the OS provides these. Furthermore, GNU `find` allows multiple types to be specified at once in a comma-separated list.

-ok Supported. Interpretation of the response is according to the 'yes' and 'no' patterns selected by setting the 'LC_MESSAGES' environment variable. When the 'POSIXLY_CORRECT' environment variable is set, these patterns are taken system's definition of a positive (yes) or negative (no) response. See the system's documentation for `nl_langinfo(3)`, in particular YESEXPR and NOEXPR. When 'POSIXLY_CORRECT' is not set, the patterns are instead taken from `find`'s own message catalogue.

-newer

Supported. If the file specified is a symbolic link, it is always dereferenced. This is a change from previous behaviour, which used to take the relevant time from the symbolic link; see the HISTORY section below.

-perm Supported. If the `POSIXLY_CORRECT` environment variable is not set, some mode arguments (for example +a+x) which are not valid in POSIX are supported for backward-compatibility.

Other primaries

The primaries **-atime**, **-ctime**, **-depth**, **-exec**, **-group**, **-links**, **-mtime**, **-nogroup**, **-nouser**, **-ok**, **-path**, **-print**, **-prune**, **-size**, **-user** and **-xdev** are all supported.

The POSIX standard specifies parentheses '(', ')' , negation '!' and the logical AND/OR operators **-a** and **-o**.

All other options, predicates, expressions and so forth are extensions beyond the POSIX standard. Many of these extensions are not unique to GNU `find`, however.

The POSIX standard requires that `find` detects loops:

The `find` utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, `find` shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

GNU `find` complies with these requirements. The link count of directories which contain entries which are hard links to an ancestor will often be lower than they otherwise should be. This can mean that GNU `find` will sometimes optimise away the visiting of a subdirectory which is actually a link to an ancestor. Since `find` does not actually enter such a subdirectory, it is allowed to avoid emitting a diagnostic message. Although this behaviour may be somewhat confusing, it is unlikely that anybody actually depends on this

behaviour. If the leaf optimisation has been turned off with **-noleaf**, the directory entry will always be examined and the diagnostic message will be issued where it is appropriate. Symbolic links cannot be used to create filesystem cycles as such, but if the **-L** option or the **-follow** option is in use, a diagnostic message is issued when **find** encounters a loop of symbolic links. As with loops containing hard links, the leaf optimisation will often mean that **find** knows that it doesn't need to call *stat()* or *chdir()* on the symbolic link, so this diagnostic is frequently not necessary.

The **-d** option is supported for compatibility with various BSD systems, but you should use the POSIX-compliant option **-depth** instead.

The **POSIXLY_CORRECT** environment variable does not affect the behaviour of the **-regex** or **-iregex** tests because those tests aren't specified in the POSIX standard.

ENVIRONMENT VARIABLES

LANG Provides a default value for the internationalization variables that are unset or null.

LC_ALL

If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE

The POSIX standard specifies that this variable affects the pattern matching to be used for the **-name** option. GNU find uses the **fnmatch(3)** library function, and so support for 'LC_COLLATE' depends on the system library. This variable also affects the interpretation of the response to **-ok**; while the 'LC_MESSAGES' variable selects the actual pattern used to interpret the response to **-ok**, the interpretation of any bracket expressions in the pattern will be affected by 'LC_COLLATE'.

LC_CTYPE

This variable affects the treatment of character classes used in regular expressions and also with the **-name** test, if the system's **fnmatch(3)** library function supports this. This variable also affects the interpretation of any character classes in the regular expressions used to interpret the response to the prompt issued by **-ok**. The 'LC_CTYPE' environment variable will also affect which characters are considered to be unprintable when filenames are printed; see the section UNUSUAL FILENAMES.

LC_MESSAGES

Determines the locale to be used for internationalised messages. If the 'POSIXLY_CORRECT' environment variable is set, this also determines the interpretation of the response to the prompt made by the **-ok** action.

NLSPATH

Determines the location of the internationalisation message catalogues.

PATH Affects the directories which are searched to find the executables invoked by **-exec**, **-execdir**, **-ok** and **-okdir**.

POSIXLY_CORRECT

Determines the block size used by **-ls** and **-fls**. If **POSIXLY_CORRECT** is set, blocks are units of 512 bytes. Otherwise they are units of 1024 bytes.

Setting this variable also turns off warning messages (that is, implies **-nowarn**) by default, because POSIX requires that apart from the output for **-ok**, all messages printed on stderr are diagnostics and must result in a non-zero exit status.

When `POSIXLY_CORRECT` is not set, `-perm +zzz` is treated just like `-perm /zzz` if `+zzz` is not a valid symbolic mode. When `POSIXLY_CORRECT` is set, such constructs are treated as an error.

When `POSIXLY_CORRECT` is set, the response to the prompt made by the `-ok` action is interpreted according to the system's message catalogue, as opposed to according to `find`'s own message translations.

TZ Affects the time zone used for some of the time-related format directives of `-printf` and `-fprintf`.

EXAMPLES

Simple ‘`find|xargs`‘ approach

- Find files named `core` in or below the directory `/tmp` and delete them.

```
$ find /tmp -name core -type f -print | xargs /bin/rm -f
```

Note that this will work incorrectly if there are any filenames containing newlines, single or double quotes, or spaces.

Safer ‘`find -print0 | xargs -0`‘ approach

- Find files named `core` in or below the directory `/tmp` and delete them, processing filenames in such a way that file or directory names containing single or double quotes, spaces or newlines are correctly handled.

```
$ find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f
```

The `-name` test comes before the `-type` test in order to avoid having to call `stat(2)` on every file.

Note that there is still a race between the time `find` traverses the hierarchy printing the matching filenames, and the time the process executed by `xargs` works with that file.

Executing a command for each file

- Run `file` on every file in or below the current directory.

```
$ find . -type f -exec file '{}' \;
```

Notice that the braces are enclosed in single quote marks to protect them from interpretation as shell script punctuation. The semicolon is similarly protected by the use of a backslash, though single quotes could have been used in that case also.

In many cases, one might prefer the ‘`-exec ... +`’ or better the ‘`-execdir ... +`’ syntax for performance and security reasons.

Traversing the filesystem just once - for 2 different actions

- Traverse the filesystem just once, listing set-user-ID files and directories into `/root/suid.txt` and large files into `/root/big.txt`.

```
$ find / \
  \(-perm -4000 -fprintf /root/suid.txt '%#m %u %p\n'\) , \
  \(-size +100M -fprintf /root/big.txt '%-10s %p\n'\)
```

This example uses the line-continuation character ‘\’ on the first two lines to instruct the shell to continue reading the command on the next line.

Searching files by age

- Search for files in your home directory which have been modified in the last twenty-four hours.

```
$ find $HOME -mtime 0
```

This command works this way because the time since each file was last modified is divided by 24 hours and any remainder is discarded. That means that to match **-mtime 0**, a file will have to have a modification in the past which is less than 24 hours ago.

Searching files by permissions

- Search for files which are executable but not readable.

```
$ find /sbin /usr/sbin -executable ! -readable -print
```

- Search for files which have read and write permission for their owner, and group, but which other users can read but not write to.

```
$ find . -perm 664
```

Files which meet these criteria but have other permissions bits set (for example if someone can execute the file) will not be matched.

- Search for files which have read and write permission for their owner and group, and which other users can read, without regard to the presence of any extra permission bits (for example the executable bit).

```
$ find . -perm -664
```

This will match a file which has mode *0777*, for example.

- Search for files which are writable by somebody (their owner, or their group, or anybody else).

```
$ find . -perm /222
```

- Search for files which are writable by either their owner or their group.

```
$ find . -perm /220
$ find . -perm /u+w,g+w
$ find . -perm /u=w,g=w
```

All three of these commands do the same thing, but the first one uses the octal representation of the file mode, and the other two use the symbolic form. The files don't have to be writable by both the owner and group to be matched; either will do.

- Search for files which are writable by both their owner and their group.

```
$ find . -perm -220
$ find . -perm -g+w,u+w
```

Both these commands do the same thing.

- A more elaborate search on permissions.

```
$ find . -perm -444 -perm /222 ! -perm /111
$ find . -perm -a+r -perm /a+w ! -perm /a+x
```

These two commands both search for files that are readable for everybody (**-perm -444** or **-perm -a+r**), have at least one write bit set (**-perm /222** or **-perm /a+w**) but are not executable for anybody (**! -perm /111** or **! -perm /a+x** respectively).

Pruning - omitting files and subdirectories

- Copy the contents of */source-dir* to */dest-dir*, but omit files and directories named *.snapshot* (and anything in them). It also omits files or directories whose name ends in ‘~’, but not their contents.

```
$ cd /source-dir
$ find . -name .snapshot -prune -o !( ! -name '*~' -print0 ) \
| cpio -pmd0 /dest-dir
```

The construct **-prune -o !(... -print0)** is quite common. The idea here is that the expression before **-prune** matches things which are to be pruned. However, the **-prune** action itself returns true, so the following **-o** ensures that the right hand side is evaluated only for those directories which didn’t get pruned (the contents of the pruned directories are not even visited, so their contents are irrelevant). The expression on the right hand side of the **-o** is in parentheses only for clarity. It emphasises that the **-print0** action takes place only for things that didn’t have **-prune** applied to them. Because the default ‘and’ condition between tests binds more tightly than **-o**, this is the default anyway, but the parentheses help to show what is going on.

- Given the following directory of projects and their associated SCM administrative directories, perform an efficient search for the projects’ roots:

```
$ find repo/ \
  !( -exec test -d '{}/.svn' ';' \
  -or -exec test -d '{}/.git' ';' \
  -or -exec test -d '{}/CVS' ';' \
) -print -prune
```

Sample output:

```
repo/project1/CVS
repo/gnu/project2/.svn
repo/gnu/project3/.svn
repo/gnu/project3/src/.svn
repo/project4/.git
```

In this example, **-prune** prevents unnecessary descent into directories that have already been discovered (for example we do not search *project3/src* because we already found *project3/.svn*), but ensures sibling directories (*project2* and *project3*) are found.

Other useful examples

- Search for several file types.

```
$ find /tmp -type f,d,l
```

Search for files, directories, and symbolic links in the directory */tmp* passing these types as a comma-separated list (GNU extension), which is otherwise equivalent to the longer, yet more portable:

```
$ find /tmp !( -type f -o -type d -o -type l )
```

- Search for files with the particular name *needle* and stop immediately when we find the first one.

```
$ find / -name needle -print -quit
```

- Demonstrate the interpretation of the **%f** and **%h** format directives of the **-printf** action for some corner-cases. Here is an example including some output.

```
$ find . . / /tmp /tmp/TRACE compile compile/64/tests/find -maxdepth 0 -printf '[%h][%f]\n'
[.][.]
[.][..]
[][/]
[][tmp]
[/tmp][TRACE]
[.][compile]
[compile/64/tests][find]
```

EXIT STATUS

find exits with status 0 if all files are processed successfully, greater than 0 if errors occur. This is deliberately a very broad description, but if the return value is non-zero, you should not rely on the correctness of the results of **find**.

When some error occurs, **find** may stop immediately, without completing all the actions specified. For example, some starting points may not have been examined or some pending program invocations for **-exec ... {} +** or **-execdir ... {} +** may not have been performed.

HISTORY

As of findutils-4.2.2, shell metacharacters ('*', '?', '[' or ']' for example) used in filename patterns match a leading '.', because IEEE POSIX interpretation 126 requires this.

As of findutils-4.3.3, **-perm /000** now matches all files instead of none.

Nanosecond-resolution timestamps were implemented in findutils-4.3.3.

As of findutils-4.3.11, the **-delete** action sets **find**'s exit status to a nonzero value when it fails. However, **find** will not exit immediately. Previously, **find**'s exit status was unaffected by the failure of **-delete**.

Feature	Added in	Also occurs in
-newerXY	4.3.3	BSD
-D	4.3.1	
-O	4.3.1	
-readable	4.3.0	
-writable	4.3.0	
-executable	4.3.0	
-regextype	4.2.24	
-exec ... {} +	4.2.12	POSIX
-execdir	4.2.12	BSD
-okdir	4.2.12	
-samefile	4.2.11	
-H	4.2.5	POSIX
-L	4.2.5	POSIX
-P	4.2.5	BSD
-delete	4.2.3	
-quit	4.2.3	
-d	4.2.3	BSD
-wholename	4.2.0	
-iwholename	4.2.0	
-ignore_readdir_race	4.2.0	
-fls	4.0	
-ilname	3.8	
-iname	3.8	
-ipath	3.8	
-iregex	3.8	

The syntax **-perm +MODE** was removed in findutils-4.5.12, in favour of **-perm /MODE**. The **+MODE**

syntax had been deprecated since findutils-4.2.21 which was released in 2005.

NON-BUGS

Operator precedence surprises

The command **find . -name afile -o -name bfile -print** will never print *afile* because this is actually equivalent to **find . -name afile -o \(-name bfile -a -print\)**. Remember that the precedence of **-a** is higher than that of **-o** and when there is no operator specified between tests, **-a** is assumed.

“paths must precede expression” error message

```
$ find . -name *.c -print
find: paths must precede expression
find: possible unquoted pattern after predicate ‘-name’?
```

This happens when the shell could expand the pattern `*.c` to more than one file name existing in the current directory, and passing the resulting file names in the command line to **find** like this:

```
find . -name frcode.c locate.c word_io.c -print
```

That command is of course not going to work, because the **-name** predicate allows exactly one pattern as argument. Instead of doing things this way, you should enclose the pattern in quotes or escape the wildcard, thus allowing **find** to use the pattern with the wildcard during the search for file name matching instead of file names expanded by the parent shell:

```
$ find . -name '*.c' -print
$ find . -name '\*.c' -print
```

BUGS

There are security problems inherent in the behaviour that the POSIX standard specifies for **find**, which therefore cannot be fixed. For example, the **-exec** action is inherently insecure, and **-execdir** should be used instead.

The environment variable **LC_COLLATE** has no effect on the **-ok** action.

REPORTING BUGS

GNU findutils online help: <<https://www.gnu.org/software/findutils/#get-help>>

Report any translation bugs to <<https://translationproject.org/team/>>

Report any other issue via the form at the GNU Savannah bug tracker:

<<https://savannah.gnu.org/bugs/?group=findutils>>

General topics about the GNU findutils package are discussed at the *bug-findutils* mailing list:

<<https://lists.gnu.org/mailman/listinfo/bug-findutils>>

COPYRIGHT

Copyright © 1990-2021 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

chmod(1), locate(1), ls(1), updatedb(1), xargs(1), lstat(2), stat(2), ctime(3) fnmatch(3), printf(3), strftime(3), locatedb(5), regex(7)

Full documentation <<https://www.gnu.org/software/findutils/find>>
or available locally via: **info find**



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewall-applet

Name

firewall-applet — firewalld applet

Synopsis

```
firewall-applet [OPTIONS...]
```

Description

firewall-applet is a tray applet for firewalld.

Options

firewall-applet does not support any special options.

The following options are supported:

```
-h , --help
```

Prints a short help text and exits.

QSettings

firewall-applet has additional settings to adapt the look and feel. QSettings is used and stores them in `~/.config/firewall/applet.conf`. The file is automatically reloaded if it has been changed and the new settings will immediately be effective.

There is also the global config file `/etc/firewall/applet.conf`, which contains the default values. The settings in this file will be overloaded by settings in the user settings file.

Here is an example `applet.conf` file:

```
[General]
notifications=true
show-inactive=true
```

The following settings are supported:

notifications

The applet shows notifications if enabled. This setting can be enabled also in the applet with the "Enable Notifications" checkbox in the right mouse menu.

This setting defaults to `false`.

If notifications are shown for these actions if enabled:

- Connection to firewalld established
- Connection to firewalld lost
- Firewall has been reloaded
- Default zone has been changed
- Panic mode has been enabled or disabled
- Activation, deactivation or change of zones bound to interfaces
- Activation, deactivation or change of zones bound to sources addresses

show-inactive

Show applet also if firewalld is not running. If firewalld has been stopped or is not running the applet will be hidden and not visible in the applet tray. Enable this setting to see the applet all the time for example to be sure that the firewall is active.

This setting defaults to `false`.

shields-up

The shields-up zone name to be used if shields-up is enabled.

This setting defaults to '`block`'.

shields-down

The shields-down zone name to be used if shields-up has been deactivated again.

This setting defaults to '`public`'.

blink

If enabled, the applet icon blinks in these cases:

- Connection to firewalld lost
- Panic mode has been enabled or disabled

This setting defaults to `false`.

blink-count

The number of blinks if `blink` is enabled.

This setting defaults to `5`.

See Also

`firewall-applet(1)`, `firewalld(1)`, `firewall-cmd(1)`, `firewall-config(1)`, `firewalld.conf(5)`, `firewalld.direct(5)`,
`firewalld.dbus(5)`, `firewalld.icmptype(5)`, `firewalld.lockdown-whitelist(5)`, `firewall-offline-cmd(1)`,
`firewalld.richlanguage(5)`, `firewalld.service(5)`, `firewalld.zone(5)`, `firewalld.zones(5)`, `firewalld.policy(5)`,
`firewalld.policies(5)`, `firewalld.ipset(5)`, `firewalld.helper(5)`

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewall-cmd

Name

firewall-cmd — firewalld command line client

Synopsis

```
firewall-cmd [OPTIONS...]
```

Description

firewall-cmd is the command line client of the firewalld daemon. It provides an interface to manage the runtime and permanent configurations.

The runtime configuration in firewalld is separated from the permanent configuration. This means that things can get changed in the runtime or permanent configuration.

Options

Sequence options are the options that can be specified multiple times, the exit code is 0 if there is at least one item that succeeded. The `ALREADY_ENABLED` (11), `NOT_ENABLED` (12) and also `ZONE_ALREADY_SET` (16) errors are treated as succeeded. If there are issues while parsing the items, then these are treated as warnings and will not change the result as long as there is a succeeded one. Without any succeeded item, the exit code will depend on the error codes. If there is exactly one error code, then this is used. If there are more than one then `UNKNOWN_ERROR` (254) will be used.

The following options are supported:

General Options

`-h`, `--help`

Prints a short help text and exits.

`-V`, `--version`

Print the version string of firewalld. This option is not combinable with other options.

`-q`, `--quiet`

Do not print status messages.

Status Options

--state

Check whether the firewalld daemon is active (i.e. running). Returns an exit code 0 if it is active, `RUNNING_BUT_FAILED` if failure occurred on startup, `NOT_RUNNING` otherwise. See the section called “Exit Codes”. This will also print the state to `STDOUT`.

--reload

Reload firewall rules and keep state information. Current permanent configuration will become new runtime configuration, i.e. all runtime only changes done until reload are lost with reload if they have not been also in permanent configuration.

Note: If `FlushAllOnReload=no`, runtime changes applied via the direct interface are not affected and will therefore stay in place until firewalld daemon is restarted completely. For `FlushAllOnReload`, see [firewalld.conf\(5\)](#).

--complete-reload

Reload firewall completely, even netfilter kernel modules. This will most likely terminate active connections, because state information is lost. This option should only be used in case of severe firewall problems. For example if there are state information problems that no connection can be established with correct firewall rules.

Note: If `FlushAllOnReload=no`, runtime changes applied via the direct interface are not affected and will therefore stay in place until firewalld daemon is restarted completely. For `FlushAllOnReload`, see [firewalld.conf\(5\)](#).

--runtime-to-permanent

Save active runtime configuration and overwrite permanent configuration with it. The way this is supposed to work is that when configuring firewalld you do runtime changes only and once you're happy with the configuration and you tested that it works the way you want, you save the configuration to disk.

--check-config

Run checks on the permanent configuration. This includes XML validity and semantics.

Log Denied Options

--get-log-denied

Print the log denied setting.

--set-log-denied = `value`

Add logging rules right before reject and drop rules in the INPUT, FORWARD and OUTPUT chains for the default rules and also final reject and drop rules in zones for the configured link-

layer packet type. The possible values are: `all`, `unicast`, `broadcast`, `multicast` and `off`. The default setting is `off`, which disables the logging.

This is a runtime and permanent change and will also reload the firewall to be able to add the logging rules.

Permanent Options

`--permanent`

The permanent option `--permanent` can be used to set options permanently. These changes are not effective immediately, only after service restart/reload or system reboot. Without the `--permanent` option, a change will only be part of the runtime configuration.

If you want to make a change in runtime and permanent configuration, use the same call with and without the `--permanent` option.

The `--permanent` option can be optionally added to all options further down where it is supported.

Zone Options

`--get-default-zone`

Print default zone for connections and interfaces.

`--set-default-zone = zone`

Set default zone for connections and interfaces where no zone has been selected. Setting the default zone changes the zone for the connections or interfaces, that are using the default zone.

This is a runtime and permanent change.

`--get-active-zones`

Print currently active zones altogether with interfaces and sources used in these zones. Active zones are zones, that have a binding to an interface or source. The output format is:

```

zone1
  interfaces: interface1 interface2 ...
  sources: source1 ...
zone2
  interfaces: interface3 ...
zone3
  sources: source2 ...

```

If there are no interfaces or sources bound to the zone, the corresponding line will be omitted.

`[--permanent] --get-zones`

Print predefined zones as a space separated list.

[--permanent] --get-services

Print predefined services as a space separated list.

[--permanent] --get-icmptypes

Print predefined icmptypes as a space separated list.

[--permanent] --get-zone-of-interface = *interface*

Print the name of the zone the *interface* is bound to or no zone.

[--permanent] --get-zone-of-source = *source* [/ *mask*] || *MAC* |ipset: *ipset*

Print the name of the zone the source is bound to or no zone.

[--permanent] --info-zone= *zone*

Print information about the zone *zone*. The output format is:

```
zone
  interfaces: interface1 ...
  sources: source1 ...
  services: service1 ...
  ports: port1 ...
  protocols: protocol1 ...
  forward-ports:
    forward-port1
    ...
  source-ports: source-port1 ...
  icmp-blocks: icmp-type1 ...
  rich rules:
    rich-rule1
    ...
```

[--permanent] --list-all-zones

List everything added for or enabled in all zones. The output format is:

```
zone1
  interfaces: interface1 ...
  sources: source1 ...
  services: service1 ...
  ports: port1 ...
  protocols: protocol1 ...
  forward-ports:
    forward-port1
    ...
  icmp-blocks: icmp-type1 ...
  rich rules:
    rich-rule1
```

..
..
`--permanent --new-zone = zone`

Add a new permanent and empty zone.

Zone names must be alphanumeric and may additionally include characters: '_' and '-'.

`--permanent --new-zone-from-file = filename [--name = zone]`

Add a new permanent zone from a prepared zone file with an optional name override.

`--permanent --delete-zone = zone`

Delete an existing permanent zone.

`--permanent --load-zone-defaults = zone`

Load zone default settings or report NO_DEFAULTS error.

`--permanent --path-zone= zone`

Print path of the zone configuration file.

Policy Options

`[--permanent] --get-policies`

Print predefined policies as a space separated list.

`[--permanent] --info-policy = policy`

Print information about the policy `policy`.

`[--permanent] --list-all-policies`

List everything added for or enabled in all policies.

`--permanent --new-policy = policy`

Add a new permanent policy.

Policy names must be alphanumeric and may additionally include characters: '_' and '-'.

`--permanent --new-policy-from-file = filename [--name = policy]`

Add a new permanent policy from a prepared policy file with an optional name override.

`--permanent --path-policy = policy`

Print path of the policy configuration file.

`--permanent --delete-policy = policy`

Delete an existing permanent policy.

`--permanent --load-policy-defaults = policy`

Load the shipped defaults for a policy. Only applies to policies shipped with firewalld. Does not apply to user defined policies.

Options to Adapt and Query Zones and Policies

Options in this section affect only one particular zone or policy. If used with `--zone = zone` or `--policy = policy` option, they affect the specified zone or policy. If both options are omitted, they affect the default zone (see `--get-default-zone`).

`[--permanent] [--zone = zone] [--policy = policy] --list-all`

List everything added or enabled.

`--permanent [--zone = zone] [--policy = policy] --get-target`

Get the target.

`--permanent [--zone = zone] [--policy = policy] --set-target = target`

Set the target.

For zones `target` is one of: `default`, `ACCEPT`, `DROP`, `REJECT`

For policies `target` is one of: `CONTINUE`, `ACCEPT`, `DROP`, `REJECT`

`default` is similar to `REJECT`, but it implicitly allows ICMP packets.

`--permanent [--zone = zone] [--policy = policy] --set-description = description`

Set description.

`--permanent [--zone = zone] [--policy = policy] --get-description`

Print description.

`--permanent [--zone = zone] [--policy = policy] --set-short = description`

Set short description.

`--permanent [--zone = zone] [--policy = policy] --get-short`

Print short description.

`[--permanent] [--zone = zone] [--permanent] [--policy = policy] --list-services`

List services added as a space separated list.

`[--permanent] [--zone = zone] [--permanent] [--policy = policy] --add-service = service [--timeout = timeval]`

Add a service. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The service is one of the firewalld provided services. To get a list of the supported services, use **firewall-cmd --get-services**.

The `--timeout` option is not combinable with the `--permanent` option.

Note: Some services define connection tracking helpers. Helpers that may operate in client mode (e.g. tftp) must be added to an outbound policy instead of a zone to take effect for clients. Otherwise the helper will not be applied to the outbound traffic. The related traffic, as defined by the connection tracking helper, on the return path (ingress) will be allowed by the stateful firewall rules.

An example of an outbound policy for connection tracking helpers:

```
# firewall-cmd --permanent --new-policy clientConntrack
# firewall-cmd --permanent --policy clientConntrack --add-ingress-zone HOST
# firewall-cmd --permanent --policy clientConntrack --add-egress-zone ANY
# firewall-cmd --permanent --policy clientConntrack --add-service tftp

[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-
service = service
```

Remove a service. This option can be specified multiple times.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-
service = service
```

Return whether `service` has been added. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-ports
```

List ports added as a space separated list. A port is of the form `portid[-portid]/protocol`, it can be either a port and protocol pair or a port range with a protocol.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-port = portid [-
portid]/protocol [ --timeout = timeval ]
```

Add the port. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The port can either be a single port number or a port range `portid - portid`. The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

The `--timeout` option is not combinable with the `--permanent` option.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-
port = portid [-portid]/protocol
```

Remove the port. This option can be specified multiple times.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-
port = portid [- portid ]/ protocol
```

Return whether the port has been added. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-protocols
```

List protocols added as a space separated list.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-
protocol = protocol [ --timeout = timeval ]
```

Add the protocol. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards.

`timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The protocol can be any protocol supported by the system. Please have a look at `/etc/protocols` for supported protocols.

The `--timeout` option is not combinable with the `--permanent` option.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-
protocol = protocol
```

Remove the protocol. This option can be specified multiple times.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-
protocol = protocol
```

Return whether the protocol has been added. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-source-ports
```

List source ports added as a space separated list. A port is of the form `portid [- portid]/ protocol`.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-source-
port = portid [- portid ]/ protocol [ --timeout = timeval ]
```

Add the source port. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards.

`timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The port can either be a single port number or a port range `portid - portid`. The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

The `--timeout` option is not combinable with the `--permanent` option.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-source-
port = portid [- portid ]/ protocol
```

Remove the source port. This option can be specified multiple times.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-source-port = portid [- portid] / protocol
```

Return whether the source port has been added. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-icmp-blocks
```

List Internet Control Message Protocol (ICMP) type blocks added as a space separated list.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-icmp-block = icmptype [ --timeout = timeval ]
```

Add an ICMP block for `icmptype`. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The `icmptype` is the one of the icmp types firewalld supports. To get a listing of supported icmp types: **firewall-cmd --get-icmptypes**

The `--timeout` option is not combinable with the `--permanent` option.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-icmp-block = icmptype
```

Remove the ICMP block for `icmptype`. This option can be specified multiple times.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-icmp-block = icmptype
```

Return whether an ICMP block for `icmptype` has been added. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-forward-ports
```

List IPv4 forward ports added as a space separated list.

For IPv6 forward ports, please use the rich language.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-forward-port = port=portid [- portid]:proto=protocol[:toport=portid [- portid]] [:toaddr=address [/ mask]] [ --timeout = timeval ]
```

Add the IPv4 forward port. This option can be specified multiple times. If a timeout is supplied, the rule will be active for the specified amount of time and will be removed automatically afterwards. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

The port can either be a single port number `portid` or a port range `portid - portid`. The protocol can either be `tcp`, `udp`, `sctp` or `dccp`. The destination address is a simple IP address.

The `--timeout` option is not combinable with the `--permanent` option.

For *IPv6* forward ports, please use the rich language.

Note: IP forwarding will be implicitly enabled if `toaddr` is specified.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-forward-
port =port= portid [- portid ]:proto= protocol [:toport= portid [- portid ]]
[:toaddr= address [/ mask ]]
```

Remove the *IPv4* forward port. This option can be specified multiple times.

For *IPv6* forward ports, please use the rich language.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-forward-
port =port= portid [- portid ]:proto= protocol [:toport= portid [- portid ]]
[:toaddr= address [/ mask ]]
```

Return whether the *IPv4* forward port has been added. Returns 0 if true, 1 otherwise.

For *IPv6* forward ports, please use the rich language.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-masquerade [ --
timeout = timeval ]
```

Enable *IPv4* masquerade. If a timeout is supplied, masquerading will be active for the specified amount of time. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`. Masquerading is useful if the machine is a router and machines connected over an interface in another zone should be able to use the first connection.

The `--timeout` option is not combinable with the `--permanent` option.

For *IPv6* masquerading, please use the rich language.

Note: IP forwarding will be implicitly enabled.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-masquerade
```

Disable *IPv4* masquerade. If the masquerading was enabled with a timeout, it will be disabled also.

For *IPv6* masquerading, please use the rich language.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-masquerade
```

Return whether *IPv4* masquerading has been enabled. Returns 0 if true, 1 otherwise.

For *IPv6* masquerading, please use the rich language.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --list-rich-rules
```

List rich language rules added as a newline separated list.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --add-rich-
rule='rule' [ --timeout = timeval ]
```

Add rich language rule '`rule`'. This option can be specified multiple times. If a timeout is supplied, the `rule` will be active for the specified amount of time and will be removed automatically afterwards. `timeval` is either a number (of seconds) or number followed by one of characters `s` (seconds), `m` (minutes), `h` (hours), for example `20m` or `1h`.

For the rich language rule syntax, please have a look at [firewalld.richlanguage\(5\)](#).

The `--timeout` option is not combinable with the `--permanent` option.

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --remove-rich-
rule='rule'
```

Remove rich language rule '`rule`'. This option can be specified multiple times.

For the rich language rule syntax, please have a look at [firewalld.richlanguage\(5\)](#).

```
[ --permanent ] [ --zone = zone ] [ --permanent ] [ --policy = policy ] --query-rich-
rule='rule'
```

Return whether a rich language rule '`rule`' has been added. Returns 0 if true, 1 otherwise.

For the rich language rule syntax, please have a look at [firewalld.richlanguage\(5\)](#).

Options to Adapt and Query Zones

Options in this section affect only one particular zone. If used with `--zone = zone` option, they affect the specified zone. If the option is omitted, they affect default zone (see `--get-default-zone`).

```
[ --permanent ] [ --zone = zone ] --add-icmp-block-inversion
```

Enable ICMP block inversion.

```
[ --permanent ] [ --zone = zone ] --remove-icmp-block-inversion
```

Disable ICMP block inversion.

```
[ --permanent ] [ --zone = zone ] --query-icmp-block-inversion
```

Return whether ICMP block inversion is enabled. Returns 0 if true, 1 otherwise.

```
[ --permanent ] [ --zone = zone ] --add-forward
```

Enable intra zone forwarding.

```
[ --permanent ] [ --zone = zone ] --remove-forward
```

Disable intra zone forwarding.

```
[ --permanent ] [ --zone = zone ] --query-forward
```

Return whether intra zone forwarding is enabled. Returns 0 if true, 1 otherwise.

Options to Adapt and Query Policies

Options in this section affect only one particular policy. It's required to specify `--policy = policy` with these options.

`--permanent` `--policy = policy` `--get-priority`

Get the priority.

`--permanent` `--policy = policy` `--set-priority = priority`

Set the priority. The priority determines the relative ordering of policies. This is an integer value between -32768 and 32767 where -1 is the default value for new policies and 0 is reserved for internal use.

If a priority is < 0, then the policy's rules will execute before all rules in all zones.

If a priority is > 0, then the policy's rules will execute after all rules in all zones.

`[--permanent] --policy = policy` `--list-ingress-zones`

List ingress zones added as a space separated list.

`[--permanent] --policy = policy` `--add-ingress-zone = zone`

Add an ingress zone. This option can be specified multiple times.

The ingress zone is one of the firewalld provided zones or one of the pseudo-zones: HOST, ANY.

HOST is used for traffic originating from the host machine, i.e. the host running firewalld.

ANY is used for traffic originating from any zone. This can be thought of as a wild card for zones. However it does not include traffic originating from the host machine - use HOST for that.

`[--permanent] --policy = policy` `--remove-ingress-zone = zone`

Remove an ingress zone. This option can be specified multiple times.

`[--permanent] --policy = policy` `--query-ingress-zone = zone`

Return whether `zone` has been added. Returns 0 if true, 1 otherwise.

`[--permanent] --policy = policy` `--list-egress-zones`

List egress zones added as a space separated list.

`[--permanent] --policy = policy` `--add-egress-zone = zone`

Add an egress zone. This option can be specified multiple times.

The egress zone is one of the firewalld provided zones or one of the pseudo-zones: HOST, ANY.

For clarification on HOST and ANY see option `--add-ingress-zone`.

`[--permanent] --policy = policy` `--remove-egress-zone = zone`

Remove an egress zone. This option can be specified multiple times.

```
[ --permanent ] [ --policy = policy ] [ --query-egress-zone = zone ]
```

Return whether *zone* has been added. Returns 0 if true, 1 otherwise.

Options to Handle Bindings of Interfaces

Binding an interface to a zone means that this zone settings are used to restrict traffic via the interface.

Options in this section affect only one particular zone. If used with `--zone = zone` option, they affect the zone *zone*. If the option is omitted, they affect default zone (see `--get-default-zone`).

For a list of predefined zones use **firewall-cmd --get-zones**.

An interface name is a string up to 16 characters long, that may not contain ' ', '/', '!' and '*'.

```
[ --permanent ] [ --zone = zone ] [ --list-interfaces ]
```

List interfaces that are bound to zone *zone* as a space separated list. If zone is omitted, default zone will be used.

```
[ --permanent ] [ --zone = zone ] [ --add-interface = interface ]
```

Bind interface *interface* to zone *zone*. If zone is omitted, default zone will be used.

If the interface is under control of NetworkManager, it is at first connected to change the zone for the connection that is using the interface. If this fails, the zone binding is created in firewalld and the limitations below apply. For interfaces that are not under control of NetworkManager, firewalld tries to change the ZONE setting in the ifcfg file, if the file exists.

As a end user you don't need this in most cases, because NetworkManager (or legacy network service) adds interfaces into zones automatically (according to `ZONE=` option from ifcfg-*interface* file) if `NM_CONTROLLED=no` is not set. You should do it only if there's no /etc/sysconfig/network-scripts/ifcfg-*interface* file. If there is such file and you add interface to zone with this `--add-interface` option, make sure the zone is the same in both cases, otherwise the behaviour would be undefined. Please also have a look at the **firewalld(1)** man page in the `Concepts` section. For permanent association of interface with a zone, see also 'How to set or change a zone for a connection?' in **firewalld.zones(5)**.

```
[ --permanent ] [ --zone = zone ] [ --change-interface = interface ]
```

If the interface is under control of NetworkManager, it is at first connected to change the zone for the connection that is using the interface. If this fails, the zone binding is created in firewalld and the limitations below apply. For interfaces that are not under control of NetworkManager, firewalld tries to change the ZONE setting in the ifcfg file, if the file exists.

Change zone the interface `interface` is bound to to zone `zone`. It's basically `--remove-interface` followed by `--add-interface`. If the interface has not been bound to a zone before, it behaves like `--add-interface`. If zone is omitted, default zone will be used.

```
[ --permanent ] [ --zone = zone ] --query-interface = interface
```

Query whether interface `interface` is bound to zone `zone`. Returns 0 if true, 1 otherwise.

```
[ --permanent ] --remove-interface = interface
```

If the interface is under control of NetworkManager, it is at first connected to change the zone for the connection that is using the interface. If this fails, the zone binding is created in firewalld and the limitations below apply.

For the addition or change of interfaces that are not under control of NetworkManager: firewalld tries to change the ZONE setting in the ifcfg file, if an ifcfg file exists that is using the interface.

Only for the removal of interfaces that are not under control of NetworkManager: firewalld is not trying to change the ZONE setting in the ifcfg file. This is needed to make sure that an ifdown of the interface will not result in a reset of the zone setting to the default zone. Only the zone binding is then removed in firewalld then.

Remove binding of interface `interface` from zone it was previously added to.

Options to Handle Bindings of Sources

Binding a source to a zone means that this zone settings will be used to restrict traffic from this source.

A source address or address range is either an IP address or a network IP address with a mask for IPv4 or IPv6 or a MAC address or an ipset with the ipset: prefix. For IPv4, the mask can be a network mask or a plain number. For IPv6 the mask is a plain number. The use of host names is not supported.

Options in this section affect only one particular zone. If used with `--zone = zone` option, they affect the zone `zone`. If the option is omitted, they affect default zone (see `--get-default-zone`).

For a list of predefined zones use **firewall-cmd** `[--permanent] --get-zones`.

```
[ --permanent ] [ --zone = zone ] --list-sources
```

List sources that are bound to zone `zone` as a space separated list. If zone is omitted, default zone will be used.

```
[ --permanent ] [ --zone = zone ] --add-source = source [/ mask ]|MAC|ipset: ipset
```

Bind the source to zone `zone`. If zone is omitted, default zone will be used.

```
[ --zone = zone ] --change-source = source [/ mask ]|MAC|ipset: ipset
```

Change zone the source is bound to to zone `zone`. It's basically `--remove-source` followed by `--add-source`. If the source has not been bound to a zone before, it behaves like `--add-source`. If zone is omitted, default zone will be used.

`[--permanent] [--zone = zone] --query-source = source [/ mask]| MAC |ipset: ipset`

Query whether the source is bound to the zone `zone`. Returns 0 if true, 1 otherwise.

`[--permanent] --remove-source = source [/ mask]| MAC |ipset: ipset`

Remove binding of the source from zone it was previously added to.

IPSet Options

`--get-ipset-types`

Print the supported ipset types.

`--permanent --new-ipset = ipset --type = type [--family = inet | inet6] [--option = key [= value]]`

Add a new permanent and empty ipset with specifying the type and optional the family and options like `timeout`, `hashsize` and `maxelem`. For more information please have a look at `ipset(8)` man page.

ipset names must be alphanumeric and may additionally include characters: '_' and '-'.

`--permanent --new-ipset-from-file = filename [--name = ipset]`

Add a new permanent ipset from a prepared ipset file with an optional name override.

`--permanent --delete-ipset = ipset`

Delete an existing permanent ipset.

`--permanent --load-ipset-defaults = ipset`

Load ipset default settings or report NO_DEFAULTS error.

`[--permanent] --info-ipset= ipset`

Print information about the ipset `ipset`. The output format is:

```

ipset
type: type
options: option1[=value1] ...
entries: entry1 ...

```

`[--permanent] --get-ipsets`

Print predefined ipsets as a space separated list.

`--permanent --ipset = ipset --set-description = description`

Set new description to ipset

`--permanent` `--ipset = ipset` `--get-description`

Print description for ipset

`--permanent` `--ipset = ipset` `--set-short = description`

Set short description to ipset

`--permanent` `--ipset = ipset` `--get-short`

Print short description for ipset

[`--permanent`] `--ipset = ipset` `--add-entry = entry`

Add a new entry to the ipset.

Adding an entry to an ipset with option `timeout` is permitted, but these entries are not tracked by firewalld.

[`--permanent`] `--ipset = ipset` `--remove-entry = entry`

Remove an entry from the ipset.

[`--permanent`] `--ipset = ipset` `--query-entry = entry`

Return whether the entry has been added to an ipset. Returns 0 if true, 1 otherwise.

Querying an ipset with a timeout will yield an error. Entries are not tracked for ipsets with a timeout.

[`--permanent`] `--ipset = ipset` `--get-entries`

List all entries of the ipset.

[`--permanent`] `--ipset = ipset` `--add-entries-from-file = filename`

Add a new entries to the ipset from the file. For all entries that are listed in the file but already in the ipset, a warning will be printed.

The file should contain an entry per line. Lines starting with an hash or semicolon are ignored. Also empty lines.

[`--permanent`] `--ipset = ipset` `--remove-entries-from-file = filename`

Remove existing entries from the ipset from the file. For all entries that are listed in the file but not in the ipset, a warning will be printed.

The file should contain an entry per line. Lines starting with an hash or semicolon are ignored. Also empty lines.

`--permanent` `--path-ipset= ipset`

Print path of the ipset configuration file.

Service Options

Options in this section affect only one particular service.

`[--permanent] --info-service= service`

Print information about the service `service`. The output format is:

```
service
ports: port1 ..
protocols: protocol1 ..
source-ports: source-port1 ..
helpers: helper1 ..
destination: ipv1 : address1 ..
```

The following options are only usable in the permanent configuration.

`--permanent --new-service= service`

Add a new permanent and empty service.

Service names must be alphanumeric and may additionally include characters: '_' and '-'.

`--permanent --new-service-from-file= filename [--name= service]`

Add a new permanent service from a prepared service file with an optional name override.

`--permanent --delete-service= service`

Delete an existing permanent service.

`--permanent --load-service-defaults= service`

Load service default settings or report NO_DEFAULTS error.

`--permanent --path-service= service`

Print path of the service configuration file.

`--permanent --service= service --set-description= description`

Set new description to service

`--permanent --service= service --get-description`

Print description for service

`--permanent --service= service --set-short= description`

Set short description to service

`--permanent --service= service --get-short`

Print short description for service

`--permanent --service= service --add-port= portid[-portid]/protocol`

Add a new port to the permanent service.

`--permanent --service= service --remove-port= portid[-portid]/protocol`

Remove a port from the permanent service.

```
--permanent --service = service --query-port = portid[-portid]/protocol
```

Return whether the port has been added to the permanent service.

```
--permanent --service = service --get-ports
```

List ports added to the permanent service.

```
--permanent --service = service --add-protocol = protocol
```

Add a new protocol to the permanent service.

```
--permanent --service = service --remove-protocol = protocol
```

Remove a protocol from the permanent service.

```
--permanent --service = service --query-protocol = protocol
```

Return whether the protocol has been added to the permanent service.

```
--permanent --service = service --get-protocols
```

List protocols added to the permanent service.

```
--permanent --service = service --add-source-port = portid[-portid]/protocol
```

Add a new source port to the permanent service.

```
--permanent --service = service --remove-source-port = portid[-portid]/protocol
```

Remove a source port from the permanent service.

```
--permanent --service = service --query-source-port = portid[-portid]/protocol
```

Return whether the source port has been added to the permanent service.

```
--permanent --service = service --get-source-ports
```

List source ports added to the permanent service.

```
--permanent --service = service --add-helper = helper
```

Add a new helper to the permanent service.

```
--permanent --service = service --remove-helper = helper
```

Remove a helper from the permanent service.

```
--permanent --service = service --query-helper = helper
```

Return whether the helper has been added to the permanent service.

```
--permanent --service = service --get-service-helpers
```

List helpers added to the permanent service.

```
--permanent --service = service --set-destination = ipv:address[/mask]
```

Set destination for *ipv* to *address*[/*mask*] in the permanent service.

`--permanent --service = service --remove-destination = ipv`

Remove the destination for *ipv* from the permanent service.

`--permanent --service = service --query-destination = ipv : address [/ mask]`

Return whether the destination *ipv* to *address[/mask]* has been set in the permanent service.

`--permanent --service = service --get-destinations`

List destinations added to the permanent service.

`--permanent --service = service --add-include = service`

Add a new include to the permanent service.

`--permanent --service = service --remove-include = service`

Remove a include from the permanent service.

`--permanent --service = service --query-include = service`

Return whether the include has been added to the permanent service.

`--permanent --service = service --get-includes`

List includes added to the permanent service.

Helper Options

Options in this section affect only one particular helper.

`[--permanent] --info-helper= helper`

Print information about the helper *helper*. The output format is:

```
helper
family: family
module: module
ports: port1 ..
```

The following options are only usable in the permanent configuration.

`--permanent --new-helper = helper --module = nf_conntrack_module [--family = ipv4 | ipv6]`

Add a new permanent helper with module and optionally family defined.

Helper names must be alphanumeric and may additionally include characters: '-'.

`--permanent --new-helper-from-file = filename [--name = helper]`

Add a new permanent helper from a prepared helper file with an optional name override.

`--permanent --delete-helper = helper`

Delete an existing permanent helper.

`--permanent --load-helper-defaults = helper`

Load helper default settings or report NO_DEFAULTS error.

`--permanent --path-helper= helper`

Print path of the helper configuration file.

`[--permanent] --get-helpers`

Print predefined helpers as a space separated list.

`--permanent --helper = helper --set-description = description`

Set new description to helper

`--permanent --helper = helper --get-description`

Print description for helper

`--permanent --helper = helper --set-short = description`

Set short description to helper

`--permanent --helper = helper --get-short`

Print short description for helper

`--permanent --helper = helper --add-port = portid[- portid]/ protocol`

Add a new port to the permanent helper.

`--permanent --helper = helper --remove-port = portid[- portid]/ protocol`

Remove a port from the permanent helper.

`--permanent --helper = helper --query-port = portid[- portid]/ protocol`

Return whether the port has been added to the permanent helper.

`--permanent --helper = helper --get-ports`

List ports added to the permanent helper.

`--permanent --helper = helper --set-module = description`

Set module description for helper

`--permanent --helper = helper --get-module`

Print module description for helper

`--permanent --helper = helper --set-family = description`

Set family description for helper

`--permanent --helper = helper --get-family`

Print family description of helper

Internet Control Message Protocol (ICMP) type Options

Options in this section affect only one particular icmp type.

```
[ --permanent ] --info-icmptype= icmptype
```

Print information about the icmp type *icmptype*. The output format is:

```
icmptype
```

```
destination: ipv1 ..
```

The following options are only usable in the permanent configuration.

```
--permanent --new-icmptype= icmptype
```

Add a new permanent and empty icmptype.

ICMP type names must be alphanumeric and may additionally include characters: '_' and '-'.

```
--permanent --new-icmptype-from-file= filename [ --name= icmptype ]
```

Add a new permanent icmptype from a prepared icmptype file with an optional name override.

```
--permanent --delete-icmptype= icmptype
```

Delete an existing permanent icmptype.

```
--permanent --load-icmptype-defaults= icmptype
```

Load icmptype default settings or report NO_DEFAULTS error.

```
--permanent --icmptype= icmptype --set-description= description
```

Set new description to icmptype

```
--permanent --icmptype= icmptype --get-description
```

Print description for icmptype

```
--permanent --icmptype= icmptype --set-short= description
```

Set short description to icmptype

```
--permanent --icmptype= icmptype --get-short
```

Print short description for icmptype

```
--permanent --icmptype= icmptype --add-destination= ipv
```

Enable destination for ipv in permanent icmptype. ipv is one of *ipv4* or *ipv6*.

```
--permanent --icmptype= icmptype --remove-destination= ipv
```

Disable destination for ipv in permanent icmptype. ipv is one of *ipv4* or *ipv6*.

```
--permanent --icmptype= icmptype --query-destination= ipv
```

Return whether destination for ipv is enabled in permanent icmptype. ipv is one of *ipv4* or *ipv6*.

`--permanent` `--icmptype = icmptype` `--get-destinations`

List destinations in permanent icmptype.

`--permanent` `--path-icmptype= icmptype`

Print path of the icmptype configuration file.

Direct Options

DEPRECATED

The direct interface has been deprecated. It will be removed in a future release. It is superseded by policies, see [firewalld.policies\(5\)](#).

The direct options give a more direct access to the firewall. These options require user to know basic iptables concepts, i.e. `table` (filter/mangle/nat/...), `chain` (INPUT/OUTPUT/FORWARD/...), `commands` (-A/-D/-I/...), `parameters` (-p/-s/-d/-j/...) and `targets` (ACCEPT/DROP/REJECT/...).

Direct options should be used only as a last resort when it's not possible to use for example `--add-service = service` or `--add-rich-rule ='rule'`.

Warning: Direct rules behavior is different depending on the value of `FirewallBackend`. See [CAVEATS](#) in [firewalld.direct\(5\)](#).

The first argument of each option has to be `ipv4` or `ipv6` or `eb`. With `ipv4` it will be for IPv4 ([iptables\(8\)](#)), with `ipv6` for IPv6 ([ip6tables\(8\)](#)) and with `eb` for ethernet bridges ([ebtables\(8\)](#)).

`[--permanent] --direct --get-all-chains`

Get all chains added to all tables. This option concerns only chains previously added with `--direct --add-chain`.

`[--permanent] --direct --get-chains { ipv4 | ipv6 | eb } table`

Get all chains added to table `table` as a space separated list. This option concerns only chains previously added with `--direct --add-chain`.

`[--permanent] --direct --add-chain { ipv4 | ipv6 | eb } table chain`

Add a new chain with name `chain` to table `table`. Make sure there's no other chain with this name already.

There already exist basic chains to use with direct options, for example `INPUT_direct` chain (see `iptables-save | grep direct` output for all of them). These chains are jumped into before chains for zones, i.e. every rule put into `INPUT_direct` will be checked before rules in zones.

`[--permanent] --direct --remove-chain { ipv4 | ipv6 | eb } table chain`

Remove chain with name `chain` from table `table`. Only chains previously added with `--direct --add-chain` can be removed this way.

```
[ --permanent ] --direct --query-chain { ipv4 | ipv6 | eb } table chain
```

Return whether a chain with name `chain` exists in table `table`. Returns 0 if true, 1 otherwise.

This option concerns only chains previously added with `--direct --add-chain`.

```
[ --permanent ] --direct --get-all-rules
```

Get all rules added to all chains in all tables as a newline separated list of the priority and arguments. This option concerns only rules previously added with `--direct --add-rule`.

```
[ --permanent ] --direct --get-rules { ipv4 | ipv6 | eb } table chain
```

Get all rules added to chain `chain` in table `table` as a newline separated list of the priority and arguments. This option concerns only rules previously added with `--direct --add-rule`.

```
[ --permanent ] --direct --add-rule { ipv4 | ipv6 | eb } table chain priority  
args
```

Add a rule with the arguments `args` to chain `chain` in table `table` with priority `priority`.

The `priority` is used to order rules. Priority 0 means add rule on top of the chain, with a higher priority the rule will be added further down. Rules with the same priority are on the same level and the order of these rules is not fixed and may change. If you want to make sure that a rule will be added after another one, use a low priority for the first and a higher for the following.

```
[ --permanent ] --direct --remove-rule { ipv4 | ipv6 | eb } table chain priority  
args
```

Remove a rule with `priority` and the arguments `args` from chain `chain` in table `table`.

Only rules previously added with `--direct --add-rule` can be removed this way.

```
[ --permanent ] --direct --remove-rules { ipv4 | ipv6 | eb } table chain
```

Remove all rules in the chain with name `chain` exists in table `table`. This option concerns only rules previously added with `--direct --add-rule` in this chain.

```
[ --permanent ] --direct --query-rule { ipv4 | ipv6 | eb } table chain priority  
args
```

Return whether a rule with `priority` and the arguments `args` exists in chain `chain` in table `table`. Returns 0 if true, 1 otherwise. This option concerns only rules previously added with `--direct --add-rule`.

```
--direct --passthrough { ipv4 | ipv6 | eb } args
```

Pass a command through to the firewall. `args` can be all **iptables**, **ip6tables** and **ebtables** command line arguments. This command is untracked, which means that firewalld is not able to provide information about this command later on, also not a listing of the untracked passthroughs.

```
[ --permanent ] --direct --get-all-passthroughs
```

Get all passthrough rules as a newline separated list of the ipv value and arguments.

```
[ --permanent ] --direct --get-passthroughs { ipv4 | ipv6 | eb }
```

Get all passthrough rules for the ipv value as a newline separated list of the priority and arguments.

```
[ --permanent ] --direct --add-passthrough { ipv4 | ipv6 | eb } args
```

Add a passthrough rule with the arguments `args` for the ipv value.

```
[ --permanent ] --direct --remove-passthrough { ipv4 | ipv6 | eb } args
```

Remove a passthrough rule with the arguments `args` for the ipv value.

```
[ --permanent ] --direct --query-passthrough { ipv4 | ipv6 | eb } args
```

Return whether a passthrough rule with the arguments `args` exists for the ipv value. Returns 0 if true, 1 otherwise.

Lockdown Options

Local applications or services are able to change the firewall configuration if they are running as root (example: libvirt) or are authenticated using PolicyKit. With this feature administrators can lock the firewall configuration so that only applications on lockdown whitelist are able to request firewall changes.

The lockdown access check limits D-Bus methods that are changing firewall rules. Query, list and get methods are not limited.

The lockdown feature is a very light version of user and application policies for firewalld and is turned off by default.

```
--lockdown-on
```

Enable lockdown. Be careful - if firewall-cmd is not on lockdown whitelist when you enable lockdown you won't be able to disable it again with firewall-cmd, you would need to edit firewalld.conf.

This is a runtime and permanent change.

```
--lockdown-off
```

Disable lockdown.

This is a runtime and permanent change.

```
--query-lockdown
```

Query whether lockdown is enabled. Returns 0 if lockdown is enabled, 1 otherwise.

Lockdown Whitelist Options

The lockdown whitelist can contain `commands`, `contexts`, `users` and `user ids`.

If a command entry on the whitelist ends with an asterisk '*', then all command lines starting with the command will match. If the '*' is not there the absolute command inclusive arguments must match.

Command paths for users are not always the same and depends on the users PATH. Some distributions symlink **/bin** to **/usr/bin** in which case it depends on the order they appear in the PATH environment variable.

The context is the security (SELinux) context of a running application or service. To get the context of a running application use **ps -e --context**.

Warning: If the context is unconfined, then this will open access for more than the desired application.

The lockdown whitelist entries are checked in the following order:

1. `context`

2. `uid`

3. `user`

4. `command`

`[--permanent] --list-lockdown-whitelist-commands`

List all command lines that are on the whitelist.

`[--permanent] --add-lockdown-whitelist-command = command`

Add the `command` to the whitelist.

`[--permanent] --remove-lockdown-whitelist-command = command`

Remove the `command` from the whitelist.

`[--permanent] --query-lockdown-whitelist-command = command`

Query whether the `command` is on the whitelist. Returns 0 if true, 1 otherwise.

`[--permanent] --list-lockdown-whitelist-contexts`

List all contexts that are on the whitelist.

`[--permanent] --add-lockdown-whitelist-context = context`

Add the context `context` to the whitelist.

`[--permanent] --remove-lockdown-whitelist-context = context`

Remove the `context` from the whitelist.

`[--permanent] --query-lockdown-whitelist-context = context`

Query whether the `context` is on the whitelist. Returns 0 if true, 1 otherwise.

`[--permanent] --list-lockdown-whitelist-uids`

List all user ids that are on the whitelist.

`[--permanent] --add-lockdown-whitelist-uid = uid`

Add the user id `uid` to the whitelist.

`[--permanent] --remove-lockdown-whitelist-uid = uid`

Remove the user id `uid` from the whitelist.

`[--permanent] --query-lockdown-whitelist-uid = uid`

Query whether the user id `uid` is on the whitelist. Returns 0 if true, 1 otherwise.

`[--permanent] --list-lockdown-whitelist-users`

List all user names that are on the whitelist.

`[--permanent] --add-lockdown-whitelist-user = user`

Add the user name `user` to the whitelist.

`[--permanent] --remove-lockdown-whitelist-user = user`

Remove the user name `user` from the whitelist.

`[--permanent] --query-lockdown-whitelist-user = user`

Query whether the user name `user` is on the whitelist. Returns 0 if true, 1 otherwise.

Panic Options

`--panic-on`

Enable panic mode. All incoming and outgoing packets are dropped, active connections will expire. Enable this only if there are serious problems with your network environment. For example if the machine is getting hacked in.

This is a runtime only change.

`--panic-off`

Disable panic mode. After disabling panic mode established connections might work again, if panic mode was enabled for a short period of time.

This is a runtime only change.

`--query-panic`

Returns 0 if panic mode is enabled, 1 otherwise.

Examples

For more examples see <http://fedoraproject.org/wiki/FirewallID>

Example 1

Enable http service in default zone. This is runtime only change, i.e. effective until restart.

```
firewall-cmd --add-service=http
```

Example 2

Enable port 443/tcp immediately and permanently in default zone. To make the change effective immediately and also after restart we need two commands. The first command makes the change in runtime configuration, i.e. makes it effective immediately, until restart. The second command makes the change in permanent configuration, i.e. makes it effective after restart.

```
firewall-cmd --add-port=443/tcp
firewall-cmd --permanent --add-port=443/tcp
```

Exit Codes

On success 0 is returned. On failure the output is red colored and exit code is either 2 in case of wrong command-line option usage or one of the following error codes in other cases:

String	Code
ALREADY_ENABLED	11
NOT_ENABLED	12
COMMAND_FAILED	13
NO_IPV6_NAT	14
PANIC_MODE	15
ZONE_ALREADY_SET	16
UNKNOWN_INTERFACE	17
ZONE_CONFLICT	18
BUILTIN_CHAIN	19
EBTABLES_NO_REJECT	20
NOT_OVERLOADABLE	21
NO_DEFAULTS	22
BUILTIN_ZONE	23
BUILTIN_SERVICE	24
BUILTIN_ICMPTYPE	25
NAME_CONFLICT	26

String	Code
NAME_MISMATCH	27
PARSE_ERROR	28
ACCESS_DENIED	29
UNKNOWN_SOURCE	30
RT_TO_PERM_FAILED	31
IPSET_WITH_TIMEOUT	32
BUILTIN_IPSET	33
ALREADY_SET	34
MISSING_IMPORT	35
DBUS_ERROR	36
BUILTIN_HELPER	37
NOT_APPLIED	38
INVALID_ACTION	100
INVALID_SERVICE	101
INVALID_PORT	102
INVALID_PROTOCOL	103
INVALID_INTERFACE	104
INVALID_ADDR	105
INVALID_FORWARD	106
INVALID_ICMPTYPE	107
INVALID_TABLE	108
INVALID_CHAIN	109
INVALID_TARGET	110
INVALID_IPV	111
INVALID_ZONE	112
INVALID_PROPERTY	113
INVALID_VALUE	114
INVALID_OBJECT	115
INVALID_NAME	116

String	Code
INVALID_FILENAME	117
INVALID_DIRECTORY	118
INVALID_TYPE	119
INVALID_SETTING	120
INVALID_DESTINATION	121
INVALID_RULE	122
INVALID_LIMIT	123
INVALID_FAMILY	124
INVALID_LOG_LEVEL	125
INVALID_AUDIT_TYPE	126
INVALID_MARK	127
INVALID_CONTEXT	128
INVALID_COMMAND	129
INVALID_USER	130
INVALID_UID	131
INVALID_MODULE	132
INVALID_PASSTHROUGH	133
INVALID_MAC	134
INVALID_IPSET	135
INVALID_ENTRY	136
INVALID_OPTION	137
INVALID_HELPER	138
INVALID_PRIORITY	139
INVALID_POLICY	140
INVALID_LOG_PREFIX	141
INVALID_NFLOG_GROUP	142
INVALID_NFLOG_QUEUE	143
MISSING_TABLE	200
MISSING_CHAIN	201

String	Code
MISSING_PORT	202
MISSING_PROTOCOL	203
MISSING_ADDR	204
MISSING_NAME	205
MISSING_SETTING	206
MISSING_FAMILY	207
RUNNING_BUT_FAILED	251
NOT_RUNNING	252
NOT_AUTHORIZED	253
UNKNOWN_ERROR	254

Note that return codes of **--query-*** options are special: Successful queries return 0, unsuccessful ones return 1 unless an error occurred in which case the table above applies.

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#), [firewalld.richlanguage\(5\)](#), [firewalld.service\(5\)](#), [firewalld.zone\(5\)](#), [firewalld.zones\(5\)](#), [firewalld.policy\(5\)](#), [firewalld.policies\(5\)](#), [firewalld.ipset\(5\)](#), [firewalld.helper\(5\)](#)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewall-config

Name

firewall-config — firewalld GUI configuration tool

Synopsis

```
firewall-config [OPTIONS...]
```

Description

firewall-config is a GUI configuration tool for firewalld.

Options

firewall-config does not support any special options. The only options that can be used are the general options that Gtk uses for Gtk application initialization. For more information on these options, please have a look at the runtime documentation for Gtk.

The following options are supported:

```
-h , --help
```

Prints a short help text and exits.

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#), [firewalld.richlanguage\(5\)](#), [firewalld.service\(5\)](#), [firewalld.zone\(5\)](#), [firewalld.zones\(5\)](#), [firewalld.policy\(5\)](#), [firewalld.policies\(5\)](#), [firewalld.ipset\(5\)](#), [firewalld.helper\(5\)](#)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld.conf

Name

`firewalld.conf` — firewalld configuration file

Synopsis

```
/etc/firewalld/firewalld.conf
```

Description

`firewalld.conf` is loaded by firewalld during the initialization process. The file contains the basic configuration options for firewalld.

Options

These are the options that can be set in the config file:

`DefaultZone`

This sets the default zone for connections or interfaces if the zone is not selected or specified by NetworkManager, initscripts or command line tool. The default zone is public.

`MinimalMark`

Deprecated. This option is ignored and no longer used. Marks are no longer used internally.

`CleanupModulesOnExit`

Setting this option to yes or true unloads all firewall-related kernel modules when firewalld is stopped. The default value is no or false.

`CleanupOnExit`

If firewalld stops, it cleans up all firewall rules. Setting this option to no or false leaves the current firewall rules untouched. The default value is yes or true.

`Lockdown`

If this option is enabled, firewall changes with the D-Bus interface will be limited to applications that are listed in the lockdown whitelist (see [firewalld.lockdown-whitelist\(5\)](#)). The default value is no or

false.

IPv6_rpfilter

If this option is enabled (it is by default), reverse path filter test on a packet for IPv6 is performed. If a reply to the packet would be sent via the same interface that the packet arrived on, the packet will match and be accepted, otherwise dropped. For IPv4 the rp_filter is controlled using sysctl.

Note: This feature has a performance impact. In most cases the impact is not enough to cause a noticeable difference. It requires route lookups and its execution occurs before the established connections fast path. As such it can have a significant performance impact if there is a lot of traffic. It's enabled by default for security, but can be disabled if performance is a concern.

IndividualCalls

If this option is disabled (it is by default), combined -restore calls are used and not individual calls to apply changes to the firewall. The use of individual calls increases the time that is needed to apply changes and to start the daemon, but is good for debugging as error messages are more specific.

LogDenied

Add logging rules right before reject and drop rules in the INPUT, FORWARD and OUTPUT chains for the default rules and also final reject and drop rules in zones for the configured link-layer packet type. The possible values are: `all`, `unicast`, `broadcast`, `multicast` and `off`. The default setting is `off`, which disables the logging.

AutomaticHelpers

Deprecated. This option is ignored and no longer used.

FirewallBackend

Selects the firewall backend implementation. Possible values are; `nftables` (default), or `iptables`. This applies to all firewalld primitives. The only exception is direct and passthrough rules which always use the traditional iptables, ip6tables, and ebtables backends.

Note: The iptables backend is deprecated. It will be removed in a future release.

FlushAllOnReload

Flush all runtime rules on a reload. In previous releases some runtime configuration was retained during a reload, namely; interface to zone assignment, and direct rules. This was confusing to users. To get the old behavior set this to "no". Defaults to "yes".

RFC3964_IPv4

As per RFC 3964, filter IPv6 traffic with 6to4 destination addresses that correspond to IPv4 addresses that should not be routed over the public internet. Defaults to "yes".

AllowZoneDrifting

Deprecated. This option is ignored and no longer used.

See Also

firewall-applet(1), firewalld(1), firewall-cmd(1), firewall-config(1), firewalld.conf(5), firewalld.direct(5), firewalld.dbus(5), firewalld.icmptype(5), firewalld.lockdown-whitelist(5), firewall-offline-cmd(1), firewalld.richlanguage(5), firewalld.service(5), firewalld.zone(5), firewalld.zones(5), firewalld.policy(5), firewalld.policies(5), firewalld.ipset(5), firewalld.helper(5)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld.policies

Name

`firewalld.policies` — firewalld policies

Description

What is a policy?

A policy applies a set of rules to traffic flowing between zones (see [zones\(5\)](#)). The policy affects traffic in a stateful unidirectional manner, e.g. zoneA to zoneB. This allows asynchronous filtering policies.

A policy's relationship to zones is defined by assigning a set of ingress zones and a set of egress zones. For example, if the set of ingress zones contains "public" and the set of egress zones contains "internal" then the policy will affect all traffic flowing from the "public" zone to the "internal" zone. However, since policies are unidirectional it will not apply to traffic flowing from "internal" to "public". Note that the ingress set and egress set can contain multiple zones.

Active Policies

Policies only become active if all of the following are true.

- The ingress zones list contain at least one regular zone or a single symbolic zone.
- The egress zones list contain at least one regular zone or a single symbolic zone.
- For non symbolic zones, the zone must be active. That is, it must have interfaces or sources assigned to it.

If the policy is not active then the policy has no effect.

Symbolic Zones

Regular zones are not enough to express every form of packet filtering. For example there is no zone to represent traffic flowing to or from the host running firewalld. As such, there are some symbolic zones to fill these gaps. However, symbolic zones are unique in that they're the only zone

allowed in the ingress or egress zone sets. For example, you cannot use "public" and "HOST" in the ingress zones.

Symbolic zones:

1. HOST

This symbolic zone is for traffic flowing to or from the host running firewalld. This corresponds to netfilter (iptables/nftables) chains INPUT and OUTPUT.

- If used in the egress zones list it will apply to traffic on the INPUT chain.
- If used in the ingress zones list it will apply to traffic on the OUTPUT chain.

2. ANY

This symbolic zone behaves like a wildcard for the ingress and egress zones. With the exception that it does not include "HOST". It's useful if you want a policy to apply to every zone.

- If used in the ingress zones list it will apply for traffic originating from any zone.
- If used in the egress zones list it will apply for traffic destined to any zone.

Predefined Policies

firewalld ships with some predefined policies. These may or may not be active by default. For details see the description of each policy.

- allow-host-ipv6

Similarity to Zones

Policies are similar to zones in that they are an attachment point for firewalld's primitives: services, ports, forward ports, etc. This is not a coincidence. Policies are a generalization of how zones have traditionally achieved filtering. In fact, in modern firewalld zones are internally implemented as a set of policies.

The main difference between policies and zones is that policies allow filtering in all directions: input, output, and forwarding. With a couple of exceptions zones only allow input filtering which is sufficient for an end station firewalling. However, for network level filtering or filtering on behalf of virtual machines and containers something more flexible, i.e. policies, are needed.

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#),

`firewalld.richlanguage(5)`, `firewalld.service(5)`, `firewalld.zone(5)`, `firewalld.zones(5)`, `firewalld.policy(5)`,
`firewalld.policies(5)`, `firewalld.ipset(5)`, `firewalld.helper(5)`

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld.policy

Name

firewalld.policy — firewalld policy configuration files

Synopsis

```
/etc/firewalld/policies/policy.xml
```

```
/usr/lib/firewalld/policies/policy.xml
```

Description

A firewalld policy configuration file contains the information for a policy. These are the policy descriptions, services, ports, protocols, icmp-blocks, masquerade, forward-ports and rich language rules in an XML file format. The file name has to be `policy_name.xml` where length of `policy_name` is currently limited to 17 chars.

This is the structure of a policy configuration file:

```
<?xml version="1.0" encoding="utf-8"?>
<policy [version="versionstring" ] [target="CONTINUE|ACCEPT|REJECT|DROP"]
[priority="priority"]>
  [ <ingress-zone name="zone" /> ]
  [ <egress-zone name="zone" /> ]

  [ <short>short description</short> ]
  [ <description>description</description> ]
  [ <service name="string" /> ]
  [ <port port="portid[-portid]" protocol="tcp|udp|sctp|dccp"/> ]
  [ <protocol value="protocol" /> ]
  [ <icmp-block name="string" /> ]
  [ <masquerade/> ]
  [ <forward-port port="portid[-portid]" protocol="tcp|udp|sctp|dccp" [to-
  port="portid[-portid]" ] [to-addr="IP address" /> ]
```

```
[ <source-port port="portid[-portid]" protocol="tcp|udp|sctp|dccp"/> ]
[

    <rule [family="ipv4|ipv6"] [priority="priority" ]>
        [ <source address="address[/mask] |mac="MAC" |ipset="ipset" 
[invert="True"]/> ]
            [ <destination address="address[/mask] |ipset="ipset" 
[invert="True"]/> ]
                [
                    <service name="string" /> |
                    <port port="portid[-portid]" protocol="tcp|udp|sctp|dccp"/> |
                    <protocol value="protocol" /> |
                    <icmp-block name="icmptype" /> |
                    <icmp-type name="icmptype" /> |
                    <masquerade /> |
                    <forward-port port="portid[-portid]" protocol="tcp|udp|sctp|dccp"
[to-port="portid[-portid]" ] [to-addr="address"]/>
                ]
                [
                    <log [prefix="prefix text" ]
[level="emerg|alert|crit|err|warn|notice|info|debug"]> [<limit
value="rate / duration" />] </log> |
                    <nflog [group="group id" ] [prefix="prefix text" ] [queue-
size="threshold" ]> [<limit value="rate / duration" />] </nflog>
                ]
                [
                    <audit> [<limit value="rate / duration" />] </audit> ]
                [
                    <accept> [<limit value="rate / duration" />] </accept> |
                    <reject [type="rejecttype" ]> [<limit value="rate / duration" />]
</reject> |
                    <drop> [<limit value="rate / duration" />] </drop> |
                    <mark set="mark[/mask]> [<limit value="rate / duration" />]
</mark>
                ]
            </rule>
        ]
    
```

```
</policy>
```

The config can contain these tags and attributes. Some of them are mandatory, others optional.

policy

The mandatory policy start and end tag defines the policy. This tag can only be used once in a policy configuration file. There are optional attributes for policy:

`version="string"`

To give the policy a version.

target="`CONTINUE` | `ACCEPT` | `REJECT` | `DROP`"

Can be used to accept, reject or drop every packet that doesn't match any rule (port, service, etc.). The `CONTINUE` is the default and used for policies that are non-terminal.

ingress-zone

An optional element that can be used several times. It can be the name of a firewalld zone or one of the symbolic zones: HOST, ANY. See [firewalld.policies\(5\)](#) for information about symbolic zones.

egress-zone

An optional element that can be used several times. It can be the name of a firewalld zone or one of the symbolic zones: HOST, ANY. See [firewalld.policies\(5\)](#) for information about symbolic zones.

short

Is an optional start and end tag and is used to give a more readable name.

description

Is an optional start and end tag to have a description.

service

Is an optional empty-element tag and can be used several times to have more than one service entry enabled. A service entry has exactly one attribute:

name="`string`"

The name of the service to be enabled. To get a list of valid service names `firewall-cmd --get-services` can be used.

port

Is an optional empty-element tag and can be used several times to have more than one port entry. All attributes of a port entry are mandatory:

port="`portid` [- `portid`]"

The port can either be a single port number `portid` or a port range `portid - portid`.

protocol="`tcp` | `udp` | `sctp` | `dccp`"

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

protocol

Is an optional empty-element tag and can be used several times to have more than one protocol entry. All protocol has exactly one attribute:

`value="string"`

The protocol can be any protocol supported by the system. Please have a look at </etc/protocols> for supported protocols.

icmp-block

Is an optional empty-element tag and can be used several times to have more than one icmp-block entry. Each icmp-block tag has exactly one mandatory attribute:

`name="string"`

The name of the Internet Control Message Protocol (ICMP) type to be blocked. To get a list of valid ICMP types **firewall-cmd --get-icmptypes** can be used.

tcp-mss-clamp

Is an optional empty-element tag and can be used several times. If left empty maximum segment size is set to 'pmtu'. This tag has exactly one optional attribute:

`value="string"`

Value can set maximum segment size to 'pmtu' (Path Maximum Transmission Unit) or a user-defined value that is greater than or equal to 536.

masquerade

Is an optional empty-element tag. It can be used only once. If it's present masquerading is enabled.

forward-port

Is an optional empty-element tag and can be used several times to have more than one port or packet forward entry. There are mandatory and also optional attributes for forward ports:

Mandatory attributes:

The local port and protocol to be forwarded.

`port="portid [-portid]"`

The port can either be a single port number `portid` or a port range `portid - portid`.

`protocol="tcp | udp | sctp | dccp"`

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

Optional attributes:

The destination of the forward. For local forwarding add `to-port` only. For remote forwarding add `to-addr` and use `to-port` optionally if the destination port on the destination machine should be different.

`to-port="portid [- portid]"`

The destination port or port range to forward to. If omitted, the value of the `port=` attribute will be used altogether with the `to-addr` attribute.

`to-addr="address"`

The destination IP address either for IPv4 or IPv6.

`source-port`

Is an optional empty-element tag and can be used several times to have more than one source port entry. All attributes of a source port entry are mandatory:

`port="portid [- portid]"`

The port can either be a single port number `portid` or a port range `portid - portid`.

`protocol="tcp | udp | sctp | dccp"`

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

`rule`

Is an optional element tag and can be used several times to have more than one rich language rule entry.

The general rule structure:

```
<rule [family="ipv4|ipv6"] [priority="priority"]>
    [ <source address="address [/mask]" | mac="MAC" | ipset="ipset"
[invert="True"]/> ]
    [ <destination address="address [/mask]" | ipset="ipset" [invert="True"]/>
]
    [
        <service name="string"/> |
        <port port="portid [- portid]" protocol="tcp|udp|sctp|dccp"/> |
        <protocol value="protocol"/> |
        <icmp-block name="icmptype"/> |
        <icmp-type name="icmptype"/> |
        <masquerade/> |
        <forward-port port="portid [- portid]" protocol="tcp|udp|sctp|dccp" [to-
port="portid [- portid]" [to-addr="address"]]/> |
        <source-port port="portid [- portid]" protocol="tcp|udp|sctp|dccp"/> |
    ]

```

```

<log [prefix="prefix text"] [level="emerg|alert|crit|err|warn|notice|info|debug"] > [<limit value="rate / duration" />] </log> |
    <nflog [group="group id"] [prefix="prefix text"] [queue-size="threshold"]> [<limit value="rate / duration" />] </nflog>
]
[ <audit> [<limit value="rate / duration" />] </audit> ]
[
    <accept> [<limit value="rate / duration" />] </accept> |
    <reject [type="rejecttype"]> [<limit value="rate / duration" />]
</reject> |
    <drop> [<limit value="rate / duration" />] </drop> |
    <mark set="mark [ / mask]"> [<limit value="rate / duration" />] </mark>
]
</rule>

```

Rule structure for source black or white listing:

```

<rule [family="ipv4|ipv6"] [priority="priority"]>
    <source address="address [ / mask] | mac="MAC" | ipset="ipset" [invert="True" ]/>
    [
        <log [prefix="prefix text"] [level="emerg|alert|crit|err|warn|notice|info|debug"] > [<limit value="rate / duration" />] </log> |
            <nflog [group="group id"] [prefix="prefix text"] [queue-size="threshold"]> [<limit value="rate / duration" />] </nflog>
        ]
        [ <audit> [<limit value="rate / duration" />] </audit> ]
        <accept> [<limit value="rate / duration" />] </accept> |
        <reject [type="rejecttype"]> [<limit value="rate / duration" />] </reject> |
        <drop> [<limit value="rate / duration" />] </drop>
    ]
</rule>

```

For a full description on rich language rules, please have a look at [firewalld.richlanguage\(5\)](#).

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#), [firewalld.richlanguage\(5\)](#), [firewalld.service\(5\)](#), [firewalld.zone\(5\)](#), [firewalld.zones\(5\)](#), [firewalld.policy\(5\)](#), [firewalld.policies\(5\)](#), [firewalld.ipset\(5\)](#), [firewalld.helper\(5\)](#)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld.service

Name

firewalld.service — firewalld service configuration files

Synopsis

```
/etc/firewalld/services/service.xml  
/usr/lib/firewalld/services/service.xml
```

Description

A firewalld service configuration file provides the information of a service entry for firewalld. The most important configuration options are ports, modules and destination addresses.

This example configuration file shows the structure of a service configuration file:

```
<?xml version="1.0" encoding="utf-8"?>  
<service>  
  <short> My Service </short>  
  <description> description </description>  
  <port port=" 137 " protocol=" tcp " />  
  <protocol value=" igmp " />  
  <module name=" nf_conntrack_netbios_ns " />  
  <destination ipv4=" 224.0.0.251 " ipv6=" ff02::fb " />  
  <include service=" ssdp " />  
  <helper name=" ftp " />  
</service>
```

Options

The config can contain these tags and attributes. Some of them are mandatory, others optional.

service

The mandatory service start and end tag defines the service. This tag can only be used once in a service configuration file. There are optional attributes for services:

`version="string"`

To give the service a version.

short

Is an optional start and end tag and is used to give an service a more readable name.

description

Is an optional start and end tag to have a description for a service.

port

Is an optional empty-element tag and can be used several times to have more than one port entry.

All attributes of a port entry are mandatory:

`port="string"`

The port `string` can be a single port number or a port range `portid - portid` or also empty to match a protocol only.

`protocol="string"`

The protocol value can either be `tcp`, `udp`, `sctp` or `dccp`.

For compatibility with older firewalld versions, it is possible to add protocols with the port option where the port is empty. With the addition of native protocol support in the service, this is not needed anymore. These entries will automatically be converted to protocols. With the next modification of the service file, the entries will be listed as protocols.

protocol

Is an optional empty-element tag and can be used several times to have more than one protocol entry. A protocol entry has exactly one attribute:

`value="string"`

The protocol can be any protocol supported by the system. Please have a look at `/etc/protocols` for supported protocols.

source-port

Is an optional empty-element tag and can be used several times to have more than one source port entry. All attributes of a source port entry are mandatory:

`port="string"`

The port `string` can be a single port number or a port range `portid - portid`.

`protocol="string"`

The protocol value can either be `tcp`, `udp`, `sctp` or `dccp`.

module

This element is deprecated. Please use helper described below in **the section called “helper”**.

destination

Is an optional empty-element tag and can be used only once. The destination specifies the destination network as a network IP address (optional with /mask), or a plain IP address. The use of hostnames is not recommended, because these will only be resolved at service activation and transmitted to the kernel. For more information in this element, please have a look at `--destination` in [iptables\(8\)](#) and [ip6tables\(8\)](#).

`ipv4="address [/mask]"`

The IPv4 destination address with optional mask.

`ipv6="address [/mask]"`

The IPv6 destination address with optional mask.

include

Is an optional empty-element tag and can be used several times to have more than one include entry. An include entry has exactly one attribute:

`service="string"`

The include can be any service supported by firewalld.

Warning:Firewalld will only check that the included service is a valid service if it's applied to a zone.

helper

Is an optional empty-element tag and can be used several times to have more than one helper entry. An helper entry has exactly one attribute:

`name="string"`

The helper can be any helper supported by firewalld.

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#), [firewalld.richlanguage\(5\)](#), [firewalld.service\(5\)](#), [firewalld.zone\(5\)](#), [firewalld.zones\(5\)](#), [firewalld.policy\(5\)](#), [firewalld.policies\(5\)](#), [firewalld.ipset\(5\)](#), [firewalld.helper\(5\)](#)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld.zone

Name

firewalld.zone — firewalld zone configuration files

Synopsis

```
/etc/firewalld/zones/zone.xml
```

```
/usr/lib/firewalld/zones/zone.xml
```

Description

A firewalld zone configuration file contains the information for a zone. These are the zone description, services, ports, protocols, icmp-blocks, masquerade, forward-ports, intra-zone forwarding and rich language rules in an XML file format. The file name has to be `zone_name.xml` where length of `zone_name` is currently limited to 17 chars.

This is the structure of a zone configuration file:

```
<?xml version="1.0" encoding="utf-8"?>
<zone [version="versionstring" ] [target="ACCEPT|%%REJECT%%|DROP"]>
    [ <interface name="string" /> ]
    [ <source address="address [/mask] " |mac="MAC" |ipset="ipset" /> ]
    [ <icmp-block-inversion/> ]
    [ <forward/> ]

    [ <short> short description </short> ]
    [ <description> description </description> ]
    [ <service name="string" /> ]
    [ <port port="portid [-portid] " protocol="tcp|udp|sctp|dccp"/> ]
    [ <protocol value="protocol" /> ]
    [ <icmp-block name="string" /> ]
    [ <masquerade/> ]
    [ <forward-port port="portid [-portid] " protocol="tcp|udp|sctp|dccp" [to-
```

```

port=" [portid][-portid]" ] [to-addr=" [IP address]" ]/> ]
[ <source-port port=" [portid][-portid]" protocol="tcp|udp|sctp|dccp"/> ]
[
    <rule [family="ipv4|ipv6"] [priority=" priority "]>
        [ <source address=" address [/ mask]" |mac=" MAC " |ipset=" ipset "
[invert=" True "]/> ]
            [ <destination address=" address [/ mask]" |ipset=" ipset "
[invert=" True "]/> ]
                [
                    <service name=" string "/> |
                    <port port=" [portid][-portid]" protocol="tcp|udp|sctp|dccp"/> |
                    <protocol value=" protocol "/> |
                    <icmp-block name=" icmptype "/> |
                    <icmp-type name=" icmptype "/> |
                    <masquerade/> |
                    <forward-port port=" [portid][-portid]" protocol="tcp|udp|sctp|dccp"
[to-port=" [portid][-portid]" ] [to-addr=" address "]/>
                ]
                [
                    <log [prefix=" prefix text "]
[level="emerg|alert|crit|err|warn|notice|info|debug"]> [<limit
value=" rate / duration " />] </log> |
                        <nflog [group=" group id "] [prefix=" prefix text "] [queue-
size=" threshold "]> [<limit value=" rate / duration " />] </nflog>
                    ]
                    [ <audit> [<limit value=" rate / duration " />] </audit> ]
                    [
                        <accept> [<limit value=" rate / duration " />] </accept> |
                        <reject [type=" rejecttype "]> [<limit value=" rate / duration " />]
</reject> |
                            <drop> [<limit value=" rate / duration " />] </drop> |
                            <mark set=" mark [/ mask ]"> [<limit value=" rate / duration " />]
</mark>
                        ]
                    </rule>
                ]
            
```

</zone>

The config can contain these tags and attributes. Some of them are mandatory, others optional.

zone

The mandatory zone start and end tag defines the zone. This tag can only be used once in a zone configuration file. There are optional attributes for zones:

`version="string"`

To give the zone a version.

`target="ACCEPT | %%REJECT%% | DROP"`

Can be used to accept, reject or drop every packet that doesn't match any rule (port, service, etc.). The `ACCEPT` target is used in `trusted` zone to accept every packet not matching any rule. The `%%REJECT%%` target is used in `block` zone to reject (with default firewalld reject type) every packet not matching any rule. The `DROP` target is used in `drop` zone to drop every packet not matching any rule. If the target is not specified, every packet not matching any rule will be rejected.

interface

Is an optional empty-element tag and can be used several times. It can be used to bind an interface to a zone. You don't need this for NetworkManager-managed interfaces, because NetworkManager binds interfaces to zones automatically. See also 'How to set or change a zone for a connection?' in [firewalld.zones\(5\)](#). You can use it as a fallback mechanism for interfaces that can't be managed via NetworkManager. An interface entry has exactly one attribute:

`name="string"`

The name of the interface to be bound to the zone.

source

Is an optional empty-element tag and can be used several times. It can be used to bind a source address, address range, a MAC address or an ipset to a zone. A source entry has exactly one of these attributes:

`address="address [/mask]"`

The source is either an IP address or a network IP address with a mask for IPv4 or IPv6. The network family (IPv4/IPv6) will be automatically discovered. For IPv4, the mask can be a network mask or a plain number. For IPv6 the mask is a plain number. The use of host names is not supported.

`mac="MAC"`

The source is a MAC address. It must be of the form XX:XX:XX:XX:XX:XX.

`ipset="ipset"`

The source is an ipset.

icmp-block-inversion

Is an optional empty-element tag and can be used only once in a zone configuration. This flag inverts the icmp block handling. Only enabled ICMP types are accepted and all others are rejected in the zone.

forward

Is an optional empty-element tag and can be used only once in a zone configuration. This flag enables intra-zone forwarding. When enabled, packets will be forwarded between interfaces or sources within a zone, even if the zone's target is not set to `ACCEPT`.

short

Is an optional start and end tag and is used to give a more readable name.

description

Is an optional start and end tag to have a description.

service

Is an optional empty-element tag and can be used several times to have more than one service entry enabled. A service entry has exactly one attribute:

`name="string"`

The name of the service to be enabled. To get a list of valid service names `firewall-cmd --get-services` can be used.

port

Is an optional empty-element tag and can be used several times to have more than one port entry. All attributes of a port entry are mandatory:

`port="portid [-portid]"`

The port can either be a single port number `portid` or a port range `portid - portid`.

`protocol="tcp | udp | sctp | dccp"`

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

protocol

Is an optional empty-element tag and can be used several times to have more than one protocol entry. All protocol has exactly one attribute:

`value="string"`

The protocol can be any protocol supported by the system. Please have a look at `/etc/protocols` for supported protocols.

icmp-block

Is an optional empty-element tag and can be used several times to have more than one icmp-block entry. Each icmp-block tag has exactly one mandatory attribute:

`name="string"`

The name of the Internet Control Message Protocol (ICMP) type to be blocked. To get a list of valid ICMP types **firewall-cmd --get-icmptypes** can be used.

tcp-mss-clamp

Is an optional empty-element tag and can be used several times. If left empty maximum segment size is set to 'pmtu'. This tag has exactly one optional attribute:

`value="string"`

Value can set maximum segment size to 'pmtu' (Path Maximum Transmission Unit) or a user-defined value that is greater than or equal to 536.

masquerade

Is an optional empty-element tag. It can be used only once. If it's present masquerading is enabled.

forward-port

Is an optional empty-element tag and can be used several times to have more than one port or packet forward entry. There are mandatory and also optional attributes for forward ports:

Mandatory attributes:

The local port and protocol to be forwarded.

`port="portid [-portid]"`

The port can either be a single port number `portid` or a port range `portid - portid`.

`protocol="tcp | udp | sctp | dccp"`

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

Optional attributes:

The destination of the forward. For local forwarding add `to-port` only. For remote forwarding add `to-addr` and use `to-port` optionally if the destination port on the destination machine should be different.

`to-port="portid [-portid]"`

The destination port or port range to forward to. If omitted, the value of the `port=` attribute will be used altogether with the `to-addr` attribute.

`to-addr="address"`

The destination IP address either for IPv4 or IPv6.

source-port

Is an optional empty-element tag and can be used several times to have more than one source port entry. All attributes of a source port entry are mandatory:

`port="portid [-portid]"`

The port can either be a single port number `portid` or a port range `portid - portid`.

`protocol="tcp | udp | sctp | dccp"`

The protocol can either be `tcp`, `udp`, `sctp` or `dccp`.

rule

Is an optional element tag and can be used several times to have more than one rich language rule entry.

The general rule structure:

```
<rule [family="ipv4|ipv6"] [priority="priority"]>
    [ <source address="address [/mask]" | mac="MAC" | ipset="ipset"
    [invert="True"] /> ]
    [ <destination address="address [/mask]" | ipset="ipset" [invert="True"] />
    ]
    [
        <service name="string" /> |
        <port port="portid [-portid]" protocol="tcp|udp|sctp|dccp" /> |
        <protocol value="protocol" /> |
        <icmp-block name="icmptype" /> |
        <icmp-type name="icmptype" /> |
        <masquerade /> |
        <forward-port port="portid [-portid]" protocol="tcp|udp|sctp|dccp" [to-
port="portid [-portid]" ] [to-addr="address"] /> |
        <source-port port="portid [-portid]" protocol="tcp|udp|sctp|dccp" /> |
    ]
    [
        <log [prefix="prefix text"] [level="emerg|alert|crit|err|warn|notice|info|debug"]> [<limit
value="rate / duration" />] </log> |
        <nflog [group="group id"] [prefix="prefix text"] [queue-
size="threshold"]> [<limit value="rate / duration" />] </nflog>
    ]
    [ <audit> [<limit value="rate / duration" />] </audit> ]
    [
        <accept> [<limit value="rate / duration" />] </accept> |
        <reject [type="rejecttype"]> [<limit value="rate / duration" />]
```

```

</reject> |
    <drop> [<limit value="rate / duration" />] </drop> |
    <mark set="mark [/ mask" /> [<limit value="rate / duration" />] </mark>
]
</rule>

```

Rule structure for source black or white listing:

```

<rule [family="ipv4|ipv6"] [priority="priority"]>
    <source address="address [/ mask" |mac="MAC" |ipset="ipset"|
[invert="True"]/>
    [
        <log [prefix="prefix text"]|
[level="emerg|alert|crit|err|warn|notice|info|debug"]> [<limit
value="rate / duration" />] </log> |
        <nflog [group="group id" ] [prefix="prefix text"] [queue-
size="threshold" ]> [<limit value="rate / duration" />] </nflog>
    ]
    [ <audit> [<limit value="rate / duration" />] </audit> ]
    <accept> [<limit value="rate / duration" />] </accept> |
    <reject [type="rejecttype"]> [<limit value="rate / duration" />] </reject> |
    <drop> [<limit value="rate / duration" />] </drop>
</rule>

```

For a full description on rich language rules, please have a look at [firewalld.richlanguage\(5\)](#).

See Also

[firewall-applet\(1\)](#), [firewalld\(1\)](#), [firewall-cmd\(1\)](#), [firewall-config\(1\)](#), [firewalld.conf\(5\)](#), [firewalld.direct\(5\)](#), [firewalld.dbus\(5\)](#), [firewalld.icmptype\(5\)](#), [firewalld.lockdown-whitelist\(5\)](#), [firewall-offline-cmd\(1\)](#), [firewalld.richlanguage\(5\)](#), [firewalld.service\(5\)](#), [firewalld.zone\(5\)](#), [firewalld.zones\(5\)](#), [firewalld.policy\(5\)](#), [firewalld.policies\(5\)](#), [firewalld.ipset\(5\)](#), [firewalld.helper\(5\)](#)

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallD>

All website content subject to the [Unlicense](#).



A service daemon with D-Bus interface

Documentation > Manual Pages >

firewalld

Name

firewalld — Dynamic Firewall Manager

Synopsis

```
firewalld [OPTIONS...]
```

Description

firewalld provides a dynamically managed firewall with support for network/firewall zones to define the trust level of network connections or interfaces. It has support for IPv4, IPv6 firewall settings and for ethernet bridges and has a separation of runtime and permanent configuration options. It also supports an interface for services or applications to add firewall rules directly.

Options

These are the command line options of firewalld:

```
-h , --help
```

Prints a short help text and exits.

```
--default-config
```

Path to firewalld default configuration. This usually defaults to `/usr/lib/firewalld`.

```
--debug [= level]
```

Set the debug level for firewalld to `level`. The range of the debug level is 1 (lowest level) to 10 (highest level). The debug output will be written to the firewalld log file specified by `--log-file`.

```
--debug-gc
```

Print garbage collector leak information. The collector runs every 10 seconds and if there are leaks, it prints information about the leaks.

```
--log-target
```

Define the output target to which log messages are written. In mixed mode, Firewalld writes info-level log messages to syslog. Debug messages are written to a file (see the `--log-file` parameter).

Info messages also go to stdout and stderr. The syslog, file or console modes write all messages to the one configured target only.

--log-file

Define the file where debug messages are written to. The default file is `/var/log/firewalld`.

--nofork

Turn off daemon forking. Force firewalld to run as a foreground process instead of as a daemon in the background.

--nopid

Disable writing pid file. By default the program will write a pid file. If the program is invoked with this option it will not check for an existing server process.

--system-config

Path to firewalld system (user) configuration. This usually defaults to `/etc/firewalld`.

Concepts

firewalld has a D-Bus interface for firewall configuration of services and applications. It also has a command line client for the user. Services or applications already using D-Bus can request changes to the firewall with the D-Bus interface directly. For more information on the firewalld D-Bus interface, please have a look at [firewalld.dbus\(5\)](#).

firewalld provides support for zones, predefined services and ICMP types and has a separation of runtime and permanent configuration options. Permanent configuration is loaded from XML files in `/usr/lib/firewalld` (`--default-config`) or `/etc/firewalld` (`--system-config`) (see [the section called “Directories”](#)).

If NetworkManager is not in use and firewalld gets started after the network is already up, the connections and manually created interfaces are not bound to the zone specified in the ifcfg file. The interfaces will automatically be handled by the default zone. firewalld will also not get notified about network device renames. All this also applies to interfaces that are not controlled by NetworkManager if `NM_CONTROLLED=no` is set.

You can add these interfaces to a zone with `firewall-cmd --permanent --zone=zone --add-interface=interface`. If there is a `/etc/sysconfig/network-scripts/ifcfg-interface` file, firewalld tries to change the `ZONE=zone` setting in this file.

If firewalld gets reloaded, it will restore the interface bindings that were in place before reloading to keep interface bindings stable in the case of NetworkManager uncontrolled interfaces. This mechanism is not possible in the case of a firewalld service restart.

It is essential to keep the `ZONE=` setting in the ifcfg file consistent to the binding in firewalld in the case of NetworkManager uncontrolled interfaces.

Zones

A network or firewall zone defines the trust level of the interface used for a connection. There are several pre-defined zones provided by firewalld. Zone configuration options and generic information about zones are described in [firewalld.zone\(5\)](#)

Services

A service can be a list of local ports, protocols and destinations and additionally also a list of firewall helper modules automatically loaded if a service is enabled. Service configuration options and generic information about services are described in [firewalld.service\(5\)](#). The use of predefined services makes it easier for the user to enable and disable access to a service.

ICMP types

The Internet Control Message Protocol (ICMP) is used to exchange information and also error messages in the Internet Protocol (IP). ICMP types can be used in firewalld to limit the exchange of these messages. For more information, please have a look at [firewalld.icmptype\(5\)](#).

Runtime configuration

Runtime configuration is the actual active configuration and is not permanent. After reload/restart of the service or a system reboot, runtime settings will be gone if they haven't been also in permanent configuration.

Permanent configuration

The permanent configuration is stored in config files and will be loaded and become new runtime configuration with every machine boot or service reload/restart.

Direct interface

DEPRECATED

The direct interface has been deprecated. It will be removed in a future release. It is superseded by policies, see [firewalld.policies\(5\)](#).

The direct interface is mainly used by services or applications to add specific firewall rules. It requires basic knowledge of ip(6)tables concepts (tables, chains, commands, parameters, targets).

Directories

firewalld supports two configuration directories:

Default/Fallback configuration in `/usr/lib/firewalld` (`--default-config`)

This directory contains the default and fallback configuration provided by firewalld for icmptypes, services and zones. The files provided with the firewalld package should not get changed and the changes are gone with an update of the firewalld package. Additional `icmptypes`, `services` and `zones` can be provided with packages or by creating files.

System configuration settings in `/etc/firewalld` (`--system-config`)

The system or user configuration stored here is either created by the system administrator or by customization with the configuration interface of firewalld or by hand. The files will overload the default configuration files.

To manually change settings of pre-defined icmptypes, zones or services, copy the file from the default configuration directory to the corresponding directory in the system configuration directory and change it accordingly.

For more information on icmptypes, please have a look at the `firewalld.icmptype(5)` man page, for services at `firewalld.service(5)` and for zones at `firewalld.zone(5)`.

SIGNALS

Currently only SIGHUP is supported.

SIGHUP

Reloads the complete firewall configuration. You can also use `firewall-cmd --reload`. All runtime configuration settings will be restored. Permanent configuration will change according to options defined in the configuration files.

See Also

`firewall-applet(1)`, `firewalld(1)`, `firewall-cmd(1)`, `firewall-config(1)`, `firewalld.conf(5)`, `firewalld.direct(5)`, `firewalld.dbus(5)`, `firewalld.icmptype(5)`, `firewalld.lockdown-whitelist(5)`, `firewall-offline-cmd(1)`, `firewalld.richlanguage(5)`, `firewalld.service(5)`, `firewalld.zone(5)`, `firewalld.zones(5)`, `firewalld.policy(5)`, `firewalld.policies(5)`, `firewalld.ipset(5)`, `firewalld.helper(5)`

Notes

firewalld home page:

<http://firewalld.org>

More documentation with examples:

<http://fedoraproject.org/wiki/FirewallID>

All website content subject to the [Unlicense](#).

NAME

free – Display amount of free and used memory in the system

SYNOPSIS

free [*options*]

DESCRIPTION

free displays the total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel. The information is gathered by parsing /proc/meminfo. The displayed columns are:

total Total installed memory (MemTotal and SwapTotal in /proc/meminfo)
used Used memory (calculated as **total** - **free** - **buffers** - **cache**)
free Unused memory (MemFree and SwapFree in /proc/meminfo)
shared Memory used (mostly) by tmpfs (Shmem in /proc/meminfo)
buffers Memory used by kernel buffers (Buffers in /proc/meminfo)
cache Memory used by the page cache and slabs (Cached and SReclaimable in /proc/meminfo)
buff/cache

Sum of **buffers** and **cache**

available

Estimation of how much memory is available for starting new applications, without swapping. Unlike the data provided by the **cache** or **free** fields, this field takes into account page cache and also that not all reclaimable memory slabs will be reclaimed due to items being in use (MemAvailable in /proc/meminfo, available on kernels 3.14, emulated on kernels 2.6.27+, otherwise the same as **free**)

OPTIONS**-b, --bytes**

Display the amount of memory in bytes.

-k, --kibi

Display the amount of memory in kibibytes. This is the default.

-m, --mebi

Display the amount of memory in mebibytes.

-g, --gibi

Display the amount of memory in gibibytes.

--tebi

Display the amount of memory in tebibytes.

--pebi

Display the amount of memory in pebibytes.

--kilo

Display the amount of memory in kilobytes. Implies --si.

--mega

Display the amount of memory in megabytes. Implies --si.

--giga

Display the amount of memory in gigabytes. Implies --si.

--tera

Display the amount of memory in terabytes. Implies --si.

--peta

Display the amount of memory in petabytes. Implies --si.

-h, --human

Show all output fields automatically scaled to shortest three digit unit and display the units of print out. Following units are used.

B = bytes

Ki = kibibyte

Mi = mebibyte

Gi = gibibyte
Ti = tebibyte
Pi = pebibyte

If unit is missing, and you have exbibyte of RAM or swap, the number is in tebibytes and columns might not be aligned with header.

-w, --wide

Switch to the wide mode. The wide mode produces lines longer than 80 characters. In this mode **buffers** and **cache** are reported in two separate columns.

-c, --count *count*

Display the result *count* times. Requires the **-s** option.

-l, --lohi

Show detailed low and high memory statistics.

-s, --seconds *delay*

Continuously display the result *delay* seconds apart. You may actually specify any floating point number for *delay* using either . or , for decimal point. **usleep(3)** is used for microsecond resolution delay times.

--si Use kilo, mega, giga etc (power of 1000) instead of kibi, mebi, gibi (power of 1024).**-t, --total**

Display a line showing the column totals.

--help Print help.**-V, --version**

Display version information.

FILES

/proc/meminfo
memory information

BUGS

The value for the **shared** column is not available from kernels before 2.6.32 and is displayed as zero.

Please send bug reports to
[<procps@freelists.org>](mailto:procps@freelists.org)

SEE ALSO

ps(1), slabtop(1), top(1), vmstat(8).

NAME

malloc, free, calloc, realloc, reallocarray – allocate and free dynamic memory

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <stdlib.h>
void *malloc(size_t size);
void free(void *ptr);
void *calloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);
void *reallocarray(void *ptr, size_t nmemb, size_t size);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
reallocarray():
Since glibc 2.29:
    _DEFAULT_SOURCE
glibc 2.28 and earlier:
    _GNU_SOURCE
```

DESCRIPTION**malloc()**

The **malloc()** function allocates *size* bytes and returns a pointer to the allocated memory. *The memory is not initialized.* If *size* is 0, then **malloc()** returns a unique pointer value that can later be successfully passed to **free()**. (See "Nonportable behavior" for portability issues.)

free()

The **free()** function frees the memory space pointed to by *ptr*, which must have been returned by a previous call to **malloc()** or related functions. Otherwise, or if *ptr* has already been freed, undefined behavior occurs. If *ptr* is NULL, no operation is performed.

calloc()

The **calloc()** function allocates memory for an array of *nmemb* elements of *size* bytes each and returns a pointer to the allocated memory. The memory is set to zero. If *nmemb* or *size* is 0, then **calloc()** returns a unique pointer value that can later be successfully passed to **free()**.

If the multiplication of *nmemb* and *size* would result in integer overflow, then **calloc()** returns an error. By contrast, an integer overflow would not be detected in the following call to **malloc()**, with the result that an incorrectly sized block of memory would be allocated:

```
malloc(nmemb * size);
```

realloc()

The **realloc()** function changes the size of the memory block pointed to by *ptr* to *size* bytes. The contents of the memory will be unchanged in the range from the start of the region up to the minimum of the old and new sizes. If the new size is larger than the old size, the added memory will *not* be initialized.

If *ptr* is NULL, then the call is equivalent to **malloc(size)**, for all values of *size*.

If *size* is equal to zero, and *ptr* is not NULL, then the call is equivalent to **free(ptr)** (but see "Nonportable behavior" for portability issues).

Unless *ptr* is NULL, it must have been returned by an earlier call to **malloc** or related functions. If the area pointed to was moved, a **free(ptr)** is done.

reallocarray()

The **reallocarray()** function changes the size of (and possibly moves) the memory block pointed to by *ptr* to be large enough for an array of *nmemb* elements, each of which is *size* bytes. It is equivalent to the call

```
realloc(ptr, nmemb * size);
```

However, unlike that **realloc()** call, **reallocarray()** fails safely in the case where the multiplication would

overflow. If such an overflow occurs, **reallocarray()** returns an error.

RETURN VALUE

The **malloc()**, **calloc()**, **realloc()**, and **reallocarray()** functions return a pointer to the allocated memory, which is suitably aligned for any type that fits into the requested size or less. On error, these functions return NULL and set *errno*. Attempting to allocate more than **PTRDIFF_MAX** bytes is considered an error, as an object that large could cause later pointer subtraction to overflow.

The **free()** function returns no value, and preserves *errno*.

The **realloc()** and **reallocarray()** functions return NULL if *ptr* is not NULL and the requested size is zero; this is not considered an error. (See "Nonportable behavior" for portability issues.) Otherwise, the returned pointer may be the same as *ptr* if the allocation was not moved (e.g., there was room to expand the allocation in-place), or different from *ptr* if the allocation was moved to a new address. If these functions fail, the original block is left untouched; it is not freed or moved.

ERRORS

calloc(), **malloc()**, **realloc()**, and **reallocarray()** can fail with the following error:

ENOMEM

Out of memory. Possibly, the application hit the **RLIMIT_AS** or **RLIMIT_DATA** limit described in **getrlimit(2)**.

VERSIONS

reallocarray() was added in glibc 2.26.

malloc() and related functions rejected sizes greater than **PTRDIFF_MAX** starting in glibc 2.30.

free() preserved *errno* starting in glibc 2.33.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
malloc() , free() , calloc() , realloc()	Thread safety	MT-Safe

STANDARDS

malloc(), **free()**, **calloc()**, **realloc()**: POSIX.1-2001, POSIX.1-2008, C99.

reallocarray() is a nonstandard extension that first appeared in OpenBSD 5.6 and FreeBSD 11.0.

NOTES

By default, Linux follows an optimistic memory allocation strategy. This means that when **malloc()** returns non-NULL there is no guarantee that the memory really is available. In case it turns out that the system is out of memory, one or more processes will be killed by the OOM killer. For more information, see the description of */proc/sys/vm/overcommit_memory* and */proc/sys/vm/oom_adj* in **proc(5)**, and the Linux kernel source file *Documentation/vm/overcommit-accounting.rst*.

Normally, **malloc()** allocates memory from the heap, and adjusts the size of the heap as required, using **sbrk(2)**. When allocating blocks of memory larger than **MMAP_THRESHOLD** bytes, the glibc **malloc()** implementation allocates the memory as a private anonymous mapping using **mmap(2)**. **MMAP_THRESHOLD** is 128 kB by default, but is adjustable using **mallopt(3)**. Prior to Linux 4.7 allocations performed using **mmap(2)** were unaffected by the **RLIMIT_DATA** resource limit; since Linux 4.7, this limit is also enforced for allocations performed using **mmap(2)**.

To avoid corruption in multithreaded applications, mutexes are used internally to protect the memory-management data structures employed by these functions. In a multithreaded application in which threads simultaneously allocate and free memory, there could be contention for these mutexes. To scalably handle memory allocation in multithreaded applications, glibc creates additional *memory allocation arenas* if mutex contention is detected. Each arena is a large region of memory that is internally allocated by the system (using **brk(2)** or **mmap(2)**), and managed with its own mutexes.

If your program uses a private memory allocator, it should do so by replacing **malloc()**, **free()**, **calloc()**, and

realloc(). The replacement functions must implement the documented glibc behaviors, including *errno* handling, size-zero allocations, and overflow checking; otherwise, other library routines may crash or operate incorrectly. For example, if the replacement *free()* does not preserve *errno*, then seemingly unrelated library routines may fail without having a valid reason in *errno*. Private memory allocators may also need to replace other glibc functions; see "Replacing malloc" in the glibc manual for details.

Crashes in memory allocators are almost always related to heap corruption, such as overflowing an allocated chunk or freeing the same pointer twice.

The **malloc()** implementation is tunable via environment variables; see **mallopt(3)** for details.

Nonportable behavior

The behavior of these functions when the requested size is zero is glibc specific; other implementations may return NULL without setting *errno*, and portable POSIX programs should tolerate such behavior. See **realloc(3p)**.

POSIX requires memory allocators to set *errno* upon failure. However, the C standard does not require this, and applications portable to non-POSIX platforms should not assume this.

Portable programs should not use private memory allocators, as POSIX and the C standard do not allow replacement of **malloc()**, **free()**, **calloc()**, and **realloc()**.

EXAMPLES

```
#include <err.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MALLOCARRAY(n, type) ((type *) my_mallocarray(n, sizeof(type)))
#define MALLOC(type)          MALLOCARRAY(1, type)

static inline void *my_mallocarray(size_t nmemb, size_t size);

int
main(void)
{
    char *p;

    p = MALLOCARRAY(32, char);
    if (p == NULL)
        err(EXIT_FAILURE, "malloc");

    strlcpy(p, "foo", 32);
    puts(p);
}

static inline void *
my_mallocarray(size_t nmemb, size_t size)
{
    return reallocarray(NULL, nmemb, size);
}
```

SEE ALSO

valgrind(1), **brk(2)**, **mmap(2)**, **alloca(3)**, **malloc_get_state(3)**, **malloc_info(3)**, **malloc_trim(3)**, **malloc_usable_size(3)**, **mallopt(3)**, **mcheck(3)**, **mtrace(3)**, **posix_memalign(3)**

For details of the GNU C library implementation, see <<https://sourceware.org/glibc/wiki/MallocInternals>>.

NAME

alloc_hugepages, free_hugepages – allocate or free huge pages

SYNOPSIS

```
void *syscall(SYS_alloc_hugepages, int key, void *addr[.len], size_t len,
              int prot, int flag);
int syscall(SYS_free_hugepages, void *addr);
```

Note: glibc provides no wrappers for these system calls, necessitating the use of **syscall**(2).

DESCRIPTION

The system calls **alloc_hugepages()** and **free_hugepages()** were introduced in Linux 2.5.36 and removed again in Linux 2.5.54. They existed only on i386 and ia64 (when built with **CONFIG_HUGETLB_PAGE**). In Linux 2.4.20, the syscall numbers exist, but the calls fail with the error **ENOSYS**.

On i386 the memory management hardware knows about ordinary pages (4 KiB) and huge pages (2 or 4 MiB). Similarly ia64 knows about huge pages of several sizes. These system calls serve to map huge pages into the process's memory or to free them again. Huge pages are locked into memory, and are not swapped.

The *key* argument is an identifier. When zero the pages are private, and not inherited by children. When positive the pages are shared with other applications using the same *key*, and inherited by child processes.

The *addr* argument of **free_hugepages()** tells which page is being freed: it was the return value of a call to **alloc_hugepages()**. (The memory is first actually freed when all users have released it.) The *addr* argument of **alloc_hugepages()** is a hint, that the kernel may or may not follow. Addresses must be properly aligned.

The *len* argument is the length of the required segment. It must be a multiple of the huge page size.

The *prot* argument specifies the memory protection of the segment. It is one of **PR_O_T_READ**, **PROT_WRITE**, **PROT_EXEC**.

The *flag* argument is ignored, unless *key* is positive. In that case, if *fla g* is **IPC_CREAT**, then a new huge page segment is created when none with the given key existed. If this flag is not set, then **ENOENT** is returned when no segment with the given key exists.

RETURN VALUE

On success, **alloc_hugepages()** returns the allocated virtual address, and **free_hugepages()** returns zero. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS**ENOSYS**

The system call is not supported on this kernel.

FILES

/proc/sys/vm/nr_hugepages

Number of configured hugetlb pages. This can be read and written.

/proc/meminfo

Gives info on the number of configured hugetlb pages and on their size in the three variables **HugePages_Total**, **HugePages_Free**, **Hugepagesize**.

STANDARDS

These extinct system calls were specific to Linux on Intel processors.

NOTES

These system calls are gone; they existed only in Linux 2.5.36 through to Linux 2.5.54. Now the hugetlfs filesystem can be used instead. Memory backed by huge pages (if the CPU supports them) is obtained by using **mmap**(2) to map files in this virtual filesystem.

The maximal number of huge pages can be specified using the **hugepages=** boot parameter.

NAME

fsck – check and repair a Linux filesystem

SYNOPSIS

fsck [**-lsAVRTMNP**] [**-r [fd]**] [**-C [fd]**] [**-t fstype**] [*filesystem...*] [**--**] [*fs-specific-options*]

DESCRIPTION

fsck is used to check and optionally repair one or more Linux filesystems. *filesystem* can be a device name (e.g., */dev/hdc1*, */dev/sdb2*), a mount point (e.g., */*, */usr*, */home*), or an filesystem label or UUID specifier (e.g., *UUID=868abf6-88c5-4a83-98b8-bfc24057f7bd* or *LABEL=root*). Normally, the **fsck** program will try to handle filesystems on different physical disk drives in parallel to reduce the total amount of time needed to check all of them.

If no filesystems are specified on the command line, and the **-A** option is not specified, **fsck** will default to checking filesystems in */etc/fstab* serially. This is equivalent to the **-As** options.

The exit status returned by **fsck** is the sum of the following conditions:

0	No errors
1	Filesystem errors corrected
2	System should be rebooted
4	Filesystem errors left uncorrected
8	Operational error
16	Usage or syntax error
32	Checking canceled by user request
128	Shared-library error

The exit status returned when multiple filesystems are checked is the bit-wise OR of the exit statuses for each filesystem that is checked.

In actuality, **fsck** is simply a front-end for the various filesystem checkers (**fsck.fstype**) available under Linux. The filesystem-specific checker is searched for in the **PATH** environment variable. If the **PATH** is undefined then fallback to */sbin*.

Please see the filesystem-specific checker manual pages for further details.

OPTIONS

-l

Create an exclusive **flock**(2) lock file (*/run/fsck/<diskname>.lock*) for whole-disk device. This option can be used with one device only (this means that **-A** and **-l** are mutually exclusive). This option is recommended when more **fsck** instances are executed in the same time. The option is ignored when

used for multiple devices or for non-rotating disks. **fsck** does not lock underlying devices when executed to check stacked devices (e.g. MD or DM) – this feature is not implemented yet.

-r [fd]

Report certain statistics for each **fsck** when it completes. These statistics include the exit status, the maximum run set size (in kilobytes), the elapsed all-clock time and the user and system CPU time used by the **fsck** run. For example:

```
/dev/sda1: status 0, rss 92828, real 4.002804, user 2.677592, sys 0.86186
```

GUI front-ends may specify a file descriptor *fd*, in which case the progress bar information will be sent to that file descriptor in a machine parseable format. For example:

```
/dev/sda1 0 92828 4.002804 2.677592 0.86186
```

-s

Serialize **fsck** operations. This is a good idea if you are checking multiple filesystems and the checkers are in an interactive mode. (Note: **e2fsck(8)** runs in an interactive mode by default. To make **e2fsck(8)** run in a non-interactive mode, you must either specify the **-p** or **-a** option, if you wish for errors to be corrected automatically, or the **-n** option if you do not.)

-t fslist

Specifies the type(s) of filesystem to be checked. When the **-A** flag is specified, only filesystems that match *fslist* are checked. The *fslist* parameter is a comma-separated list of filesystems and options specifiers. All of the filesystems in this comma-separated list may be prefixed by a negation operator '**no**' or '**!**', which requests that only those filesystems not listed in *fslist* will be checked. If none of the filesystems in *fslist* is prefixed by a negation operator, then only those listed filesystems will be checked.

Options specifiers may be included in the comma-separated *fslist*. They must have the format **opts=fs-option**. If an options specifier is present, then only filesystems which contain *fs-option* in their mount options field of */etc/fstab* will be checked. If the options specifier is prefixed by a negation operator, then only those filesystems that do not have *fs-option* in their mount options field of */etc/fstab* will be checked.

For example, if **opts=ro** appears in *fslist*, then only filesystems listed in */etc/fstab* with the **ro** option will be checked.

For compatibility with Mandrake distributions whose boot scripts depend upon an unauthorized UI change to the **fsck** program, if a filesystem type of **loop** is found in *fslist*, it is treated as if **opts=loop** were specified as an argument to the **-t** option.

Normally, the filesystem type is deduced by searching for *filesys* in the */etc/fstab* file and using the corresponding entry. If the type cannot be deduced, and there is only a single filesystem given as an argument to the **-t** option, **fsck** will use the specified filesystem type. If this type is not available, then the default filesystem type (currently ext2) is used.

-A

Walk through the */etc/fstab* file and try to check all filesystems in one run. This option is typically used from the */etc/rc* system initialization file, instead of multiple commands for checking a single filesystem.

The root filesystem will be checked first unless the **-P** option is specified (see below). After that, filesystems will be checked in the order specified by the *fs_passno* (the sixth) field in the */etc/fstab* file.

Filesystems with a *fs_passno* value of 0 are skipped and are not checked at all. Filesystems with a *fs_passno* value of greater than zero will be checked in order, with filesystems with the lowest *fs_passno* number being checked first. If there are multiple filesystems with the same pass number, **fsck** will attempt to check them in parallel, although it will avoid running multiple filesystem checks on the same physical disk.

fsck does not check stacked devices (RAIDs, dm-crypt, ...) in parallel with any other device. See below for **FSCK_FORCE_ALL_PARALLEL** setting. The /sys filesystem is used to determine dependencies between devices.

Hence, a very common configuration in /etc/fstab files is to set the root filesystem to have a *fs_passno* value of 1 and to set all other filesystems to have a *fs_passno* value of 2. This will allow **fsck** to automatically run filesystem checkers in parallel if it is advantageous to do so. System administrators might choose not to use this configuration if they need to avoid multiple filesystem checks running in parallel for some reason – for example, if the machine in question is short on memory so that excessive paging is a concern.

fsck normally does not check whether the device actually exists before calling a filesystem specific checker. Therefore non-existing devices may cause the system to enter filesystem repair mode during boot if the filesystem specific checker returns a fatal error. The /etc/fstab mount option **nofail** may be used to have **fsck** skip non-existing devices. **fsck** also skips non-existing devices that have the special filesystem type **auto**.

-C [fd]

Display completion/progress bars for those filesystem checkers (currently only for ext[234]) which support them. **fsck** will manage the filesystem checkers so that only one of them will display a progress bar at a time. GUI front-ends may specify a file descriptor *fd*, in which case the progress bar information will be sent to that file descriptor.

-M

Do not check mounted filesystems and return an exit status of 0 for mounted filesystems.

-N

Don't execute, just show what would be done.

-P

When the **-A** flag is set, check the root filesystem in parallel with the other filesystems. This is not the safest thing in the world to do, since if the root filesystem is in doubt things like the **e2fsck(8)** executable might be corrupted! This option is mainly provided for those sysadmins who don't want to repartition the root filesystem to be small and compact (which is really the right solution).

-R

When checking all filesystems with the **-A** flag, skip the root filesystem. (This is useful in case the root filesystem has already been mounted read-write.)

-T

Don't show the title on startup.

-V

Produce verbose output, including all filesystem-specific commands that are executed.

-?, --help

Display help text and exit.

--version

Display version information and exit.

FILESYSTEM SPECIFIC OPTIONS

Options which are not understood by fsck are passed to the filesystem-specific checker!

These options **must** not take arguments, as there is no way for **fsck** to be able to properly guess which options take arguments and which don't.

Options and arguments which follow the **--** are treated as filesystem-specific options to be passed to the filesystem-specific checker.

Please note that **fsck** is not designed to pass arbitrarily complicated options to filesystem-specific checkers. If you're doing something complicated, please just execute the filesystem-specific checker directly. If you pass **fsck** some horribly complicated options and arguments, and it doesn't do what you expect, **don't bother reporting it as a bug**. You're almost certainly doing something that you shouldn't be doing with **fsck**. Options to different filesystem-specific **fsck**'s are not standardized.

ENVIRONMENT

The **fsck** program's behavior is affected by the following environment variables:

FSCK_FORCE_ALL_PARALLEL

If this environment variable is set, **fsck** will attempt to check all of the specified filesystems in parallel, regardless of whether the filesystems appear to be on the same device. (This is useful for RAID systems or high-end storage systems such as those sold by companies such as IBM or EMC.) Note that the *fs_passno* value is still used.

FSCK_MAX_INST

This environment variable will limit the maximum number of filesystem checkers that can be running at one time. This allows configurations which have a large number of disks to avoid **fsck** starting too many filesystem checkers at once, which might overload CPU and memory resources available on the system. If this value is zero, then an unlimited number of processes can be spawned. This is currently the default, but future versions of **fsck** may attempt to automatically determine how many filesystem checks can be run based on gathering accounting data from the operating system.

PATH

The **PATH** environment variable is used to find filesystem checkers.

FSTAB_FILE

This environment variable allows the system administrator to override the standard location of the */etc/fstab* file. It is also useful for developers who are testing **fsck**.

LIBBLKID_DEBUG=all

enables libblkid debug output.

LIBMOUNT_DEBUG=all

enables libmount debug output.

FILES

/etc/fstab

AUTHORS

[Theodore Ts'o <tytso@mit.edu>](#), [Karel Zak <kzak@redhat.com>](#)

SEE ALSO

[fstab\(5\)](#), [mkfs\(8\)](#), [fsck.ext2\(8\)](#) or [fsck.ext3\(8\)](#) or [e2fsck\(8\)](#), [fsck.cramfs\(8\)](#), [fsck.jfs\(8\)](#), [fsck.nfs\(8\)](#), [fsck.minix\(8\)](#), [fsck.msdos\(8\)](#), [fsck.vfat\(8\)](#), [fsck.xfs\(8\)](#), [reiserfsck\(8\)](#)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **fsck** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

fsck.btrfs – do nothing, successfully

SYNOPSIS

fsck.btrfs [**-aApy**] [<*device*>...]

DESCRIPTION

fsck.btrfs is a type of utility that should exist for any filesystem and is called during system setup when the corresponding **/etc/fstab** entries contain non-zero value for **fs_passno**, see **fstab(5)** for more.

Traditional filesystems need to run their respective fsck utility in case the filesystem was not unmounted cleanly and the log needs to be replayed before mount. This is not needed for BTRFS. You should set **fs_passno** to 0.

If you wish to check the consistency of a BTRFS filesystem or repair a damaged filesystem, see **btrfs-check(8)**. By default filesystem consistency is checked, the repair mode is enabled via the **--repair** option (use with care!).

OPTIONS

The options are all the same and detect if **fsck.btrfs** is executed in non-interactive mode and exits with success, otherwise prints a message about btrfs check.

EXIT STATUS

There are two possible exit codes returned:

0

No error

8

Operational error, eg. device does not exist

FILES

/etc/fstab

SEE ALSO

btrfs(8), **fsck(8)**, **fstab(5)**

NAME

fsck.exfat – check an exFAT filesystem

SYNOPSIS

fsck.exfat [**-a**] [**-n**] [**-r**] [**-v**] [**-y**] [**-v**] *device*
fsck.exfat **-V**

DESCRIPTION

fsck.exfat checks an exFAT filesystem and repairs the filesystem depending on the options passed.

OPTIONS

- a** This option does the same thing as the **-p** option. It is provided for backwards compatibility only; it is suggested that people use **-p** option whenever possible.
- n** Check the filesystem but do not attempt to repair the filesystem.
- p** Repair the filesystem without user interaction if it can be done safely.
- r** Repair the filesystem interactively.
- v** Prints verbose debugging information while checking the exFAT filesystem.
- V** Prints the version number and exits.
- y** Repair the filesystem answering yes to all questions.

SEE ALSO

fsck(8), **fstab(5)**,

NAME

e2fsck – check a Linux ext2/ext3/ext4 file system

SYNOPSIS

```
e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ] [ -I|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [ -E extended_options ] [ -z undo_file ] device
```

DESCRIPTION

e2fsck is used to check the ext2/ext3/ext4 family of file systems. For ext3 and ext4 file systems that use a journal, if the system has been shut down uncleanly without any errors, normally, after replaying the committed transactions in the journal, the file system should be marked as clean. Hence, for file systems that use journaling, **e2fsck** will normally replay the journal and exit, unless its superblock indicates that further checking is required.

device is a block device (e.g., */dev/sdc1*) or file containing the file system.

Note that in general it is not safe to run **e2fsck** on mounted file systems. The only exception is if the **-n** option is specified, and **-c**, **-I**, or **-L** options are *not* specified. However, even if it is safe to do so, the results printed by **e2fsck** are not valid if the file system is mounted. If **e2fsck** asks whether or not you should check a file system which is mounted, the only correct answer is “no”. Only experts who really know what they are doing should consider answering this question in any other way.

If **e2fsck** is run in interactive mode (meaning that none of **-y**, **-n**, or **-p** are specified), the program will ask the user to fix each problem found in the file system. A response of ‘y’ will fix the error; ‘n’ will leave the error unfixed; and ‘a’ will fix the problem and all subsequent problems; pressing Enter will proceed with the default response, which is printed before the question mark. Pressing Control-C terminates **e2fsck** immediately.

OPTIONS

- a** This option does the same thing as the **-p** option. It is provided for backwards compatibility only; it is suggested that people use **-p** option whenever possible.
- b** *superblock*
Instead of using the normal superblock, use an alternative superblock specified by *superblock*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system’s blocksize, the number of blocks per group, and features such as **sparse_super**.
Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system’s layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).
If an alternative superblock is specified and the file system is not opened read-only, **e2fsck** will make sure that the primary superblock is updated appropriately upon completion of the file system check.
- B** *blocksize*
Normally, **e2fsck** will search for the superblock at various different block sizes in an attempt to find the appropriate block size. This search can be fooled in some cases. This option forces **e2fsck** to only try locating the superblock at a particular blocksize. If the superblock is not found, **e2fsck** will terminate with a fatal error.
- c** This option causes **e2fsck** to use **badblocks(8)** program to do a read-only scan of the device in order to find any bad blocks. If any bad blocks are found, they are added to the bad block inode to prevent them from being allocated to a file or directory. If this option is specified twice, then the bad block scan will be done using a non-destructive read-write test.
- C** *fd* This option causes **e2fsck** to write completion information to the specified file descriptor so that the progress of the file system check can be monitored. This option is typically used by programs which are running **e2fsck**. If the file descriptor number is negative, then absolute value of the file descriptor will be used, and the progress information will be suppressed initially. It can later be enabled by sending the **e2fsck** process a SIGUSR1 signal. If the file descriptor specified is 0,

e2fsck will print a completion bar as it goes about its business. This requires that e2fsck is running on a video console or terminal.

- d** Print debugging output (useless unless you are debugging **e2fsck**).
- D** Optimize directories in file system. This option causes e2fsck to try to optimize all directories, either by re-indexing them if the file system supports directory indexing, or by sorting and compressing directories for smaller directories, or for file systems using traditional linear directories.

Even without the **-D** option, **e2fsck** may sometimes optimize a few directories --- for example, if directory indexing is enabled and a directory is not indexed and would benefit from being indexed, or if the index structures are corrupted and need to be rebuilt. The **-D** option forces all directories in the file system to be optimized. This can sometimes make them a little smaller and slightly faster to search, but in practice, you should rarely need to use this option.

The **-D** option will detect directory entries with duplicate names in a single directory, which e2fsck normally does not enforce for performance reasons.

-E extended_options

Set e2fsck extended options. Extended options are comma separated, and may take an argument using the equals ('=') sign. The following options are supported:

ea_ver=extended_attribute_version

Set the version of the extended attribute blocks which **e2fsck** will require while checking the file system. The version number may be 1 or 2. The default extended attribute version format is 2.

journal_only

Only replay the journal if required, but do not perform any further checks or repairs.

fragcheck

During pass 1, print a detailed report of any discontiguous blocks for files in the file system.

discard

Attempt to discard free blocks and unused inode blocks after the full file system check (discarding blocks is useful on solid state devices and sparse / thin-provisioned storage). Note that discard is done in pass 5 AFTER the file system has been fully checked and only if it does not contain recognizable errors. However there might be cases where **e2fsck** does not fully recognize a problem and hence in this case this option may prevent you from further manual data recovery.

nodiscard

Do not attempt to discard free blocks and unused inode blocks. This option is exactly the opposite of discard option. This is set as default.

no_optimize_extents

Do not offer to optimize the extent tree by eliminating unnecessary width or depth. This can also be enabled in the options section of **/etc/e2fsck.conf**.

optimize_extents

Offer to optimize the extent tree by eliminating unnecessary width or depth. This is the default unless otherwise specified in **/etc/e2fsck.conf**.

inode_count_fullmap

Trade off using memory for speed when checking a file system with a large number of hard-linked files. The amount of memory required is proportional to the number of inodes in the file system. For large file systems, this can be gigabytes of memory. (For example, a 40TB file system with 2.8 billion inodes will consume an additional 5.7 GB memory if this optimization is

enabled.) This optimization can also be enabled in the options section of */etc/e2fsck.conf*.

no_inode_count_fullmap

Disable the **inode_count_fullmap** optimization. This is the default unless otherwise specified in */etc/e2fsck.conf*.

readahead_kb

Use this many KiB of memory to pre-fetch metadata in the hopes of reducing e2fsck runtime. By default, this is set to the size of two block groups' inode tables (typically 4MiB on a regular ext4 file system); if this amount is more than 1/50th of total physical memory, readahead is disabled. Set this to zero to disable readahead entirely.

bmap2extent

Convert block-mapped files to extent-mapped files.

fixes_only

Only fix damaged metadata; do not optimize htree directories or compress extent trees. This option is incompatible with the -D and -E bmap2extent options.

check_encoding

Force verification of encoded filenames in case-insensitive directories. This is the default mode if the file system has the strict flag enabled.

unshare_blocks

If the file system has shared blocks, with the shared blocks read-only feature enabled, then this will unshare all shared blocks and unset the read-only feature bit. If there is not enough free space then the operation will fail. If the file system does not have the read-only feature bit, but has shared blocks anyway, then this option will have no effect. Note when using this option, if there is no free space to clone blocks, there is no prompt to delete files and instead the operation will fail.

Note that **unshare_blocks** implies the "-f" option to ensure that all passes are run. Additionally, if "-n" is also specified, e2fsck will simulate trying to allocate enough space to deduplicate. If this fails, the exit code will be non-zero.

-f Force checking even if the file system seems clean.

-F Flush the file system device's buffer caches before beginning. Only really useful for doing **e2fsck** time trials.

-j *external-journal*

Set the pathname where the external-journal for this file system can be found.

-k When combined with the **-c** option, any existing bad blocks in the bad blocks list are preserved, and any new bad blocks found by running **badblocks(8)** will be added to the existing bad blocks list.

-l *filename*

Add the block numbers listed in the file specified by *filename* to the list of bad blocks. The format of this file is the same as the one generated by the **badblocks(8)** program. Note that the block numbers are based on the blocksize of the file system. Hence, **badblocks(8)** must be given the blocksize of the file system in order to obtain correct results. As a result, it is much simpler and safer to use the **-c** option to **e2fsck**, since it will assure that the correct parameters are passed to the **badblocks** program.

-L *filename*

Set the bad blocks list to be the list of blocks specified by *filename*. (This option is the same as the **-l** option, except the bad blocks list is cleared before the blocks listed in the file are added to

the bad blocks list.)

- n** Open the file system read-only, and assume an answer of ‘no’ to all questions. Allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-p** or **-y** options.
- p** Automatically repair (“preen”) the file system. This option will cause **e2fsck** to automatically fix any file system problems that can be safely fixed without human intervention. If **e2fsck** discloses a problem which may require the system administrator to take additional corrective action, **e2fsck** will print a description of the problem and then exit with the value 4 logically or’ed into the exit code. (See the **EXIT CODE** section.) This option is normally used by the system’s boot scripts. It may not be specified at the same time as the **-n** or **-y** options.
- r** This option does nothing at all; it is provided only for backwards compatibility.
- t** Print timing statistics for **e2fsck**. If this option is used twice, additional timing statistics are printed on a pass by pass basis.
- v** Verbose mode.
- V** Print version information and exit.
- y** Assume an answer of ‘yes’ to all questions; allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-n** or **-p** options.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the **undo_file** argument, the undo file will be written to a file named **e2fsck-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable.

WARNING: The undo file cannot be used to recover from a power or system crash.

EXIT CODE

The exit code returned by **e2fsck** is the sum of the following conditions:

0	– No errors
1	– File system errors corrected
2	– File system errors corrected, system should be rebooted
4	– File system errors left uncorrected
8	– Operational error
16	– Usage or syntax error
32	– E2fsck canceled by user request
128	– Shared library error

SIGNALS

The following signals have the following effect when sent to **e2fsck**.

SIGUSR1

This signal causes **e2fsck** to start displaying a completion bar or emitting progress information. (See discussion of the **-C** option.)

SIGUSR2

This signal causes **e2fsck** to stop displaying a completion bar or emitting progress information.

REPORTING BUGS

Almost any piece of software will have bugs. If you manage to find a file system which causes **e2fsck** to crash, or which **e2fsck** is unable to repair, please report it to the author.

Please include as much information as possible in your bug report. Ideally, include a complete transcript of the **e2fsck** run, so I can see exactly what error messages are displayed. (Make sure the messages printed by **e2fsck** are in English; if your system has been configured so that **e2fsck**’s messages have been translated

into another language, please set the the **LC_ALL** environment variable to **C** so that the transcript of e2fsck's output will be useful to me.) If you have a writable file system where the transcript can be stored, the **script(1)** program is a handy way to save the output of **e2fsck** to a file.

It is also useful to send the output of **dumpe2fs(8)**. If a specific inode or inodes seems to be giving **e2fsck** trouble, try running the **debugfs(8)** command and send the output of the **stat(1u)** command run on the relevant inode(s). If the inode is a directory, the **debugfs** *dump* command will allow you to extract the contents of the directory inode, which can sent to me after being first run through **uuencode(1)**. The most useful data you can send to help reproduce the bug is a compressed raw image dump of the file system, generated using **e2image(8)**. See the **e2image(8)** man page for more details.

Always include the full version string which **e2fsck** displays when it is run, so I know which version you are running.

ENVIRONMENT

E2FSCK_CONFIG

Determines the location of the configuration file (see **e2fsck.conf(5)**).

AUTHOR

This version of **e2fsck** was written by Theodore Ts'o <tytso@mit.edu>.

SEE ALSO

e2fsck.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **debugfs(8)**, **e2image(8)**, **mke2fs(8)**, **tune2fs(8)**

NAME

e2fsck – check a Linux ext2/ext3/ext4 file system

SYNOPSIS

```
e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ] [ -I|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [ -E extended_options ] [ -z undo_file ] device
```

DESCRIPTION

e2fsck is used to check the ext2/ext3/ext4 family of file systems. For ext3 and ext4 file systems that use a journal, if the system has been shut down uncleanly without any errors, normally, after replaying the committed transactions in the journal, the file system should be marked as clean. Hence, for file systems that use journaling, **e2fsck** will normally replay the journal and exit, unless its superblock indicates that further checking is required.

device is a block device (e.g., */dev/sdc1*) or file containing the file system.

Note that in general it is not safe to run **e2fsck** on mounted file systems. The only exception is if the **-n** option is specified, and **-c**, **-I**, or **-L** options are *not* specified. However, even if it is safe to do so, the results printed by **e2fsck** are not valid if the file system is mounted. If **e2fsck** asks whether or not you should check a file system which is mounted, the only correct answer is “no”. Only experts who really know what they are doing should consider answering this question in any other way.

If **e2fsck** is run in interactive mode (meaning that none of **-y**, **-n**, or **-p** are specified), the program will ask the user to fix each problem found in the file system. A response of ‘y’ will fix the error; ‘n’ will leave the error unfixed; and ‘a’ will fix the problem and all subsequent problems; pressing Enter will proceed with the default response, which is printed before the question mark. Pressing Control-C terminates **e2fsck** immediately.

OPTIONS

- a** This option does the same thing as the **-p** option. It is provided for backwards compatibility only; it is suggested that people use **-p** option whenever possible.
- b** *superblock*
Instead of using the normal superblock, use an alternative superblock specified by *superblock*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system’s blocksize, the number of blocks per group, and features such as **sparse_super**.
Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system’s layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).
If an alternative superblock is specified and the file system is not opened read-only, **e2fsck** will make sure that the primary superblock is updated appropriately upon completion of the file system check.
- B** *blocksize*
Normally, **e2fsck** will search for the superblock at various different block sizes in an attempt to find the appropriate block size. This search can be fooled in some cases. This option forces **e2fsck** to only try locating the superblock at a particular blocksize. If the superblock is not found, **e2fsck** will terminate with a fatal error.
- c** This option causes **e2fsck** to use **badblocks(8)** program to do a read-only scan of the device in order to find any bad blocks. If any bad blocks are found, they are added to the bad block inode to prevent them from being allocated to a file or directory. If this option is specified twice, then the bad block scan will be done using a non-destructive read-write test.
- C** *fd* This option causes **e2fsck** to write completion information to the specified file descriptor so that the progress of the file system check can be monitored. This option is typically used by programs which are running **e2fsck**. If the file descriptor number is negative, then absolute value of the file descriptor will be used, and the progress information will be suppressed initially. It can later be enabled by sending the **e2fsck** process a SIGUSR1 signal. If the file descriptor specified is 0,

e2fsck will print a completion bar as it goes about its business. This requires that e2fsck is running on a video console or terminal.

- d** Print debugging output (useless unless you are debugging **e2fsck**).
- D** Optimize directories in file system. This option causes e2fsck to try to optimize all directories, either by re-indexing them if the file system supports directory indexing, or by sorting and compressing directories for smaller directories, or for file systems using traditional linear directories.

Even without the **-D** option, **e2fsck** may sometimes optimize a few directories --- for example, if directory indexing is enabled and a directory is not indexed and would benefit from being indexed, or if the index structures are corrupted and need to be rebuilt. The **-D** option forces all directories in the file system to be optimized. This can sometimes make them a little smaller and slightly faster to search, but in practice, you should rarely need to use this option.

The **-D** option will detect directory entries with duplicate names in a single directory, which e2fsck normally does not enforce for performance reasons.

-E extended_options

Set e2fsck extended options. Extended options are comma separated, and may take an argument using the equals ('=') sign. The following options are supported:

ea_ver=extended_attribute_version

Set the version of the extended attribute blocks which **e2fsck** will require while checking the file system. The version number may be 1 or 2. The default extended attribute version format is 2.

journal_only

Only replay the journal if required, but do not perform any further checks or repairs.

fragcheck

During pass 1, print a detailed report of any discontiguous blocks for files in the file system.

discard

Attempt to discard free blocks and unused inode blocks after the full file system check (discarding blocks is useful on solid state devices and sparse / thin-provisioned storage). Note that discard is done in pass 5 AFTER the file system has been fully checked and only if it does not contain recognizable errors. However there might be cases where **e2fsck** does not fully recognize a problem and hence in this case this option may prevent you from further manual data recovery.

nodiscard

Do not attempt to discard free blocks and unused inode blocks. This option is exactly the opposite of discard option. This is set as default.

no_optimize_extents

Do not offer to optimize the extent tree by eliminating unnecessary width or depth. This can also be enabled in the options section of **/etc/e2fsck.conf**.

optimize_extents

Offer to optimize the extent tree by eliminating unnecessary width or depth. This is the default unless otherwise specified in **/etc/e2fsck.conf**.

inode_count_fullmap

Trade off using memory for speed when checking a file system with a large number of hard-linked files. The amount of memory required is proportional to the number of inodes in the file system. For large file systems, this can be gigabytes of memory. (For example, a 40TB file system with 2.8 billion inodes will consume an additional 5.7 GB memory if this optimization is

enabled.) This optimization can also be enabled in the options section of */etc/e2fsck.conf*.

no_inode_count_fullmap

Disable the **inode_count_fullmap** optimization. This is the default unless otherwise specified in */etc/e2fsck.conf*.

readahead_kb

Use this many KiB of memory to pre-fetch metadata in the hopes of reducing e2fsck runtime. By default, this is set to the size of two block groups' inode tables (typically 4MiB on a regular ext4 file system); if this amount is more than 1/50th of total physical memory, readahead is disabled. Set this to zero to disable readahead entirely.

bmap2extent

Convert block-mapped files to extent-mapped files.

fixes_only

Only fix damaged metadata; do not optimize htree directories or compress extent trees. This option is incompatible with the -D and -E bmap2extent options.

check_encoding

Force verification of encoded filenames in case-insensitive directories. This is the default mode if the file system has the strict flag enabled.

unshare_blocks

If the file system has shared blocks, with the shared blocks read-only feature enabled, then this will unshare all shared blocks and unset the read-only feature bit. If there is not enough free space then the operation will fail. If the file system does not have the read-only feature bit, but has shared blocks anyway, then this option will have no effect. Note when using this option, if there is no free space to clone blocks, there is no prompt to delete files and instead the operation will fail.

Note that **unshare_blocks** implies the "-f" option to ensure that all passes are run. Additionally, if "-n" is also specified, e2fsck will simulate trying to allocate enough space to deduplicate. If this fails, the exit code will be non-zero.

-f Force checking even if the file system seems clean.

-F Flush the file system device's buffer caches before beginning. Only really useful for doing **e2fsck** time trials.

-j *external-journal*

Set the pathname where the external-journal for this file system can be found.

-k When combined with the **-c** option, any existing bad blocks in the bad blocks list are preserved, and any new bad blocks found by running **badblocks(8)** will be added to the existing bad blocks list.

-l *filename*

Add the block numbers listed in the file specified by *filename* to the list of bad blocks. The format of this file is the same as the one generated by the **badblocks(8)** program. Note that the block numbers are based on the blocksize of the file system. Hence, **badblocks(8)** must be given the blocksize of the file system in order to obtain correct results. As a result, it is much simpler and safer to use the **-c** option to **e2fsck**, since it will assure that the correct parameters are passed to the **badblocks** program.

-L *filename*

Set the bad blocks list to be the list of blocks specified by *filename*. (This option is the same as the **-l** option, except the bad blocks list is cleared before the blocks listed in the file are added to

the bad blocks list.)

- n** Open the file system read-only, and assume an answer of ‘no’ to all questions. Allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-p** or **-y** options.
- p** Automatically repair (“preen”) the file system. This option will cause **e2fsck** to automatically fix any file system problems that can be safely fixed without human intervention. If **e2fsck** discovers a problem which may require the system administrator to take additional corrective action, **e2fsck** will print a description of the problem and then exit with the value 4 logically or’ed into the exit code. (See the **EXIT CODE** section.) This option is normally used by the system’s boot scripts. It may not be specified at the same time as the **-n** or **-y** options.
- r** This option does nothing at all; it is provided only for backwards compatibility.
- t** Print timing statistics for **e2fsck**. If this option is used twice, additional timing statistics are printed on a pass by pass basis.
- v** Verbose mode.
- V** Print version information and exit.
- y** Assume an answer of ‘yes’ to all questions; allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-n** or **-p** options.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the **undo_file** argument, the undo file will be written to a file named **e2fsck-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable.

WARNING: The undo file cannot be used to recover from a power or system crash.

EXIT CODE

The exit code returned by **e2fsck** is the sum of the following conditions:

0	– No errors
1	– File system errors corrected
2	– File system errors corrected, system should be rebooted
4	– File system errors left uncorrected
8	– Operational error
16	– Usage or syntax error
32	– E2fsck canceled by user request
128	– Shared library error

SIGNALS

The following signals have the following effect when sent to **e2fsck**.

SIGUSR1

This signal causes **e2fsck** to start displaying a completion bar or emitting progress information. (See discussion of the **-C** option.)

SIGUSR2

This signal causes **e2fsck** to stop displaying a completion bar or emitting progress information.

REPORTING BUGS

Almost any piece of software will have bugs. If you manage to find a file system which causes **e2fsck** to crash, or which **e2fsck** is unable to repair, please report it to the author.

Please include as much information as possible in your bug report. Ideally, include a complete transcript of the **e2fsck** run, so I can see exactly what error messages are displayed. (Make sure the messages printed by **e2fsck** are in English; if your system has been configured so that **e2fsck**’s messages have been translated

into another language, please set the the **LC_ALL** environment variable to **C** so that the transcript of e2fsck's output will be useful to me.) If you have a writable file system where the transcript can be stored, the **script(1)** program is a handy way to save the output of **e2fsck** to a file.

It is also useful to send the output of **dumpe2fs(8)**. If a specific inode or inodes seems to be giving **e2fsck** trouble, try running the **debugfs(8)** command and send the output of the **stat(1u)** command run on the relevant inode(s). If the inode is a directory, the **debugfs** *dump* command will allow you to extract the contents of the directory inode, which can sent to me after being first run through **uuencode(1)**. The most useful data you can send to help reproduce the bug is a compressed raw image dump of the file system, generated using **e2image(8)**. See the **e2image(8)** man page for more details.

Always include the full version string which **e2fsck** displays when it is run, so I know which version you are running.

ENVIRONMENT

E2FSCK_CONFIG

Determines the location of the configuration file (see **e2fsck.conf(5)**).

AUTHOR

This version of **e2fsck** was written by Theodore Ts'o <tytso@mit.edu>.

SEE ALSO

e2fsck.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **debugfs(8)**, **e2image(8)**, **mke2fs(8)**, **tune2fs(8)**

NAME

e2fsck – check a Linux ext2/ext3/ext4 file system

SYNOPSIS

```
e2fsck [ -pacnyrdfkvtDFV ] [ -b superblock ] [ -B blocksize ] [ -I|-L bad_blocks_file ] [ -C fd ] [ -j external-journal ] [ -E extended_options ] [ -z undo_file ] device
```

DESCRIPTION

e2fsck is used to check the ext2/ext3/ext4 family of file systems. For ext3 and ext4 file systems that use a journal, if the system has been shut down uncleanly without any errors, normally, after replaying the committed transactions in the journal, the file system should be marked as clean. Hence, for file systems that use journaling, **e2fsck** will normally replay the journal and exit, unless its superblock indicates that further checking is required.

device is a block device (e.g., */dev/sdc1*) or file containing the file system.

Note that in general it is not safe to run **e2fsck** on mounted file systems. The only exception is if the **-n** option is specified, and **-c**, **-I**, or **-L** options are *not* specified. However, even if it is safe to do so, the results printed by **e2fsck** are not valid if the file system is mounted. If **e2fsck** asks whether or not you should check a file system which is mounted, the only correct answer is “no”. Only experts who really know what they are doing should consider answering this question in any other way.

If **e2fsck** is run in interactive mode (meaning that none of **-y**, **-n**, or **-p** are specified), the program will ask the user to fix each problem found in the file system. A response of ‘y’ will fix the error; ‘n’ will leave the error unfixed; and ‘a’ will fix the problem and all subsequent problems; pressing Enter will proceed with the default response, which is printed before the question mark. Pressing Control-C terminates **e2fsck** immediately.

OPTIONS

- a** This option does the same thing as the **-p** option. It is provided for backwards compatibility only; it is suggested that people use **-p** option whenever possible.
- b** *superblock*
Instead of using the normal superblock, use an alternative superblock specified by *superblock*. This option is normally used when the primary superblock has been corrupted. The location of backup superblocks is dependent on the file system’s blocksize, the number of blocks per group, and features such as **sparse_super**.
Additional backup superblocks can be determined by using the **mke2fs** program using the **-n** option to print out where the superblocks exist, supposing **mke2fs** is supplied with arguments that are consistent with the file system’s layout (e.g. blocksize, blocks per group, **sparse_super**, etc.).
If an alternative superblock is specified and the file system is not opened read-only, **e2fsck** will make sure that the primary superblock is updated appropriately upon completion of the file system check.
- B** *blocksize*
Normally, **e2fsck** will search for the superblock at various different block sizes in an attempt to find the appropriate block size. This search can be fooled in some cases. This option forces **e2fsck** to only try locating the superblock at a particular blocksize. If the superblock is not found, **e2fsck** will terminate with a fatal error.
- c** This option causes **e2fsck** to use **badblocks(8)** program to do a read-only scan of the device in order to find any bad blocks. If any bad blocks are found, they are added to the bad block inode to prevent them from being allocated to a file or directory. If this option is specified twice, then the bad block scan will be done using a non-destructive read-write test.
- C** *fd* This option causes **e2fsck** to write completion information to the specified file descriptor so that the progress of the file system check can be monitored. This option is typically used by programs which are running **e2fsck**. If the file descriptor number is negative, then absolute value of the file descriptor will be used, and the progress information will be suppressed initially. It can later be enabled by sending the **e2fsck** process a SIGUSR1 signal. If the file descriptor specified is 0,

e2fsck will print a completion bar as it goes about its business. This requires that e2fsck is running on a video console or terminal.

- d** Print debugging output (useless unless you are debugging **e2fsck**).
- D** Optimize directories in file system. This option causes e2fsck to try to optimize all directories, either by re-indexing them if the file system supports directory indexing, or by sorting and compressing directories for smaller directories, or for file systems using traditional linear directories.

Even without the **-D** option, **e2fsck** may sometimes optimize a few directories --- for example, if directory indexing is enabled and a directory is not indexed and would benefit from being indexed, or if the index structures are corrupted and need to be rebuilt. The **-D** option forces all directories in the file system to be optimized. This can sometimes make them a little smaller and slightly faster to search, but in practice, you should rarely need to use this option.

The **-D** option will detect directory entries with duplicate names in a single directory, which e2fsck normally does not enforce for performance reasons.

-E extended_options

Set e2fsck extended options. Extended options are comma separated, and may take an argument using the equals ('=') sign. The following options are supported:

ea_ver=extended_attribute_version

Set the version of the extended attribute blocks which **e2fsck** will require while checking the file system. The version number may be 1 or 2. The default extended attribute version format is 2.

journal_only

Only replay the journal if required, but do not perform any further checks or repairs.

fragcheck

During pass 1, print a detailed report of any discontiguous blocks for files in the file system.

discard

Attempt to discard free blocks and unused inode blocks after the full file system check (discarding blocks is useful on solid state devices and sparse / thin-provisioned storage). Note that discard is done in pass 5 AFTER the file system has been fully checked and only if it does not contain recognizable errors. However there might be cases where **e2fsck** does not fully recognize a problem and hence in this case this option may prevent you from further manual data recovery.

nodiscard

Do not attempt to discard free blocks and unused inode blocks. This option is exactly the opposite of discard option. This is set as default.

no_optimize_extents

Do not offer to optimize the extent tree by eliminating unnecessary width or depth. This can also be enabled in the options section of **/etc/e2fsck.conf**.

optimize_extents

Offer to optimize the extent tree by eliminating unnecessary width or depth. This is the default unless otherwise specified in **/etc/e2fsck.conf**.

inode_count_fullmap

Trade off using memory for speed when checking a file system with a large number of hard-linked files. The amount of memory required is proportional to the number of inodes in the file system. For large file systems, this can be gigabytes of memory. (For example, a 40TB file system with 2.8 billion inodes will consume an additional 5.7 GB memory if this optimization is

enabled.) This optimization can also be enabled in the options section of */etc/e2fsck.conf*.

no_inode_count_fullmap

Disable the **inode_count_fullmap** optimization. This is the default unless otherwise specified in */etc/e2fsck.conf*.

readahead_kb

Use this many KiB of memory to pre-fetch metadata in the hopes of reducing e2fsck runtime. By default, this is set to the size of two block groups' inode tables (typically 4MiB on a regular ext4 file system); if this amount is more than 1/50th of total physical memory, readahead is disabled. Set this to zero to disable readahead entirely.

bmap2extent

Convert block-mapped files to extent-mapped files.

fixes_only

Only fix damaged metadata; do not optimize htree directories or compress extent trees. This option is incompatible with the -D and -E bmap2extent options.

check_encoding

Force verification of encoded filenames in case-insensitive directories. This is the default mode if the file system has the strict flag enabled.

unshare_blocks

If the file system has shared blocks, with the shared blocks read-only feature enabled, then this will unshare all shared blocks and unset the read-only feature bit. If there is not enough free space then the operation will fail. If the file system does not have the read-only feature bit, but has shared blocks anyway, then this option will have no effect. Note when using this option, if there is no free space to clone blocks, there is no prompt to delete files and instead the operation will fail.

Note that **unshare_blocks** implies the "-f" option to ensure that all passes are run. Additionally, if "-n" is also specified, e2fsck will simulate trying to allocate enough space to deduplicate. If this fails, the exit code will be non-zero.

-f Force checking even if the file system seems clean.

-F Flush the file system device's buffer caches before beginning. Only really useful for doing **e2fsck** time trials.

-j *external-journal*

Set the pathname where the external-journal for this file system can be found.

-k When combined with the **-c** option, any existing bad blocks in the bad blocks list are preserved, and any new bad blocks found by running **badblocks(8)** will be added to the existing bad blocks list.

-l *filename*

Add the block numbers listed in the file specified by *filename* to the list of bad blocks. The format of this file is the same as the one generated by the **badblocks(8)** program. Note that the block numbers are based on the blocksize of the file system. Hence, **badblocks(8)** must be given the blocksize of the file system in order to obtain correct results. As a result, it is much simpler and safer to use the **-c** option to **e2fsck**, since it will assure that the correct parameters are passed to the **badblocks** program.

-L *filename*

Set the bad blocks list to be the list of blocks specified by *filename*. (This option is the same as the **-l** option, except the bad blocks list is cleared before the blocks listed in the file are added to

the bad blocks list.)

- n** Open the file system read-only, and assume an answer of ‘no’ to all questions. Allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-p** or **-y** options.
- p** Automatically repair (“preen”) the file system. This option will cause **e2fsck** to automatically fix any file system problems that can be safely fixed without human intervention. If **e2fsck** discovers a problem which may require the system administrator to take additional corrective action, **e2fsck** will print a description of the problem and then exit with the value 4 logically or’ed into the exit code. (See the **EXIT CODE** section.) This option is normally used by the system’s boot scripts. It may not be specified at the same time as the **-n** or **-y** options.
- r** This option does nothing at all; it is provided only for backwards compatibility.
- t** Print timing statistics for **e2fsck**. If this option is used twice, additional timing statistics are printed on a pass by pass basis.
- v** Verbose mode.
- V** Print version information and exit.
- y** Assume an answer of ‘yes’ to all questions; allows **e2fsck** to be used non-interactively. This option may not be specified at the same time as the **-n** or **-p** options.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the **undo_file** argument, the undo file will be written to a file named **e2fsck-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable.

WARNING: The undo file cannot be used to recover from a power or system crash.

EXIT CODE

The exit code returned by **e2fsck** is the sum of the following conditions:

0	– No errors
1	– File system errors corrected
2	– File system errors corrected, system should be rebooted
4	– File system errors left uncorrected
8	– Operational error
16	– Usage or syntax error
32	– E2fsck canceled by user request
128	– Shared library error

SIGNALS

The following signals have the following effect when sent to **e2fsck**.

SIGUSR1

This signal causes **e2fsck** to start displaying a completion bar or emitting progress information. (See discussion of the **-C** option.)

SIGUSR2

This signal causes **e2fsck** to stop displaying a completion bar or emitting progress information.

REPORTING BUGS

Almost any piece of software will have bugs. If you manage to find a file system which causes **e2fsck** to crash, or which **e2fsck** is unable to repair, please report it to the author.

Please include as much information as possible in your bug report. Ideally, include a complete transcript of the **e2fsck** run, so I can see exactly what error messages are displayed. (Make sure the messages printed by **e2fsck** are in English; if your system has been configured so that **e2fsck**’s messages have been translated

into another language, please set the the **LC_ALL** environment variable to **C** so that the transcript of e2fsck's output will be useful to me.) If you have a writable file system where the transcript can be stored, the **script(1)** program is a handy way to save the output of **e2fsck** to a file.

It is also useful to send the output of **dumpe2fs(8)**. If a specific inode or inodes seems to be giving **e2fsck** trouble, try running the **debugfs(8)** command and send the output of the **stat(1u)** command run on the relevant inode(s). If the inode is a directory, the **debugfs** *dump* command will allow you to extract the contents of the directory inode, which can sent to me after being first run through **uuencode(1)**. The most useful data you can send to help reproduce the bug is a compressed raw image dump of the file system, generated using **e2image(8)**. See the **e2image(8)** man page for more details.

Always include the full version string which **e2fsck** displays when it is run, so I know which version you are running.

ENVIRONMENT

E2FSCK_CONFIG

Determines the location of the configuration file (see **e2fsck.conf(5)**).

AUTHOR

This version of **e2fsck** was written by Theodore Ts'o <tytso@mit.edu>.

SEE ALSO

e2fsck.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **debugfs(8)**, **e2image(8)**, **mke2fs(8)**, **tune2fs(8)**

NAME

fsck.f2fs – check a Linux F2FS file system

SYNOPSIS

```
fsck.f2fs [ -a enable auto fix ] [ -f enable force fix ] [ -p enable preen mode ] [ -t show stored directory tree ] [ -d debugging-level ] device
```

DESCRIPTION

fsck.f2fs is used to check an f2fs file system (usually in a disk partition). *device* is the special file corresponding to the device (e.g. */dev/sdXX*).

The exit code returned by **fsck.f2fs** is 0 on success and -1 on failure.

OPTIONS

-a enable auto fix

Enable to run file system check only if a bug was reported by the F2FS kernel module. It is disabled by default.

-f enable force fix

Enable to fix all the inconsistency in the partition.

-p enable preen mode

Same as "-a" to support general fsck convention.

-t show stored directory tree

Enable to show every directory entries in the partition.

-d debug-level

Specify the level of debugging options. The default number is 0, which shows basic debugging messages.

AUTHOR

Initial checking code was written by Byoung Geun Kim <bgbg.kim@samsung.com>. Jaegeuk Kim <jaegeuk@kernel.org> reworked most parts of the codes to support fixing any corrupted images.

AVAILABILITY

fsck.f2fs is available from <git://git.kernel.org/pub/scm/linux/kernel/git/jaegeuk/f2fs-tools.git>.

SEE ALSO

mkfs.f2fs(8), **dump.f2fs(8)**, **defrag.f2fs(8)**, **resize.f2fs(8)**, **sload.f2fs(8)**.

NAME

fsck.fat – check and repair MS-DOS FAT filesystems

SYNOPSIS

fsck.fat [*OPTIONS*] *DEVICE*

DESCRIPTION

fsck.fat verifies the consistency of MS-DOS filesystems and optionally tries to repair them.

The following filesystem problems can be corrected (in this order):

- FAT contains invalid cluster numbers. Cluster is changed to EOF.
- File's cluster chain contains a loop. The loop is broken.
- Bad clusters (read errors). The clusters are marked bad and they are removed from files owning them. This check is optional.
- Directories with a large number of bad entries (probably corrupt). The directory can be deleted.
- Files . and .. are non-directories. They can be deleted or renamed.
- Directories . and .. in root directory. They are deleted.
- Bad filenames. They can be renamed.
- Duplicate directory entries. They can be deleted or renamed.
- Directories with non-zero size field. Size is set to zero.
- Directory . does not point to parent directory. The start pointer is adjusted.
- Directory .. does not point to parent of parent directory. The start pointer is adjusted.
- . and .. are not the two first entries in a non-root directory. The entries are created, moving occupied slots if necessary.
- Start cluster number of a file is invalid. The file is truncated.
- File contains bad or free clusters. The file is truncated.
- File's cluster chain is longer than indicated by the size fields. The file is truncated.
- Two or more files share the same cluster(s). All but one of the files are truncated. If the file being truncated is a directory file that has already been read, the filesystem check is restarted after truncation.
- File's cluster chain is shorter than indicated by the size fields. The file is truncated.
- Volume label in root directory or label in boot sector is invalid. Invalid labels are removed.
- Volume label in root directory and label in boot sector are different. Volume label from root directory is copied to boot sector.
- Clusters are marked as used but are not owned by a file. They are marked as free.

Additionally, the following problems are detected, but not repaired:

- Invalid parameters in boot sector

When **fsck.fat** checks a filesystem, it accumulates all changes in memory and performs them only after all checks are complete. This can be disabled with the **-w** option.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

There are some minor differences in Atari format: Some boot sector fields are interpreted slightly different, and the special FAT entries for end-of-file and bad cluster can be different. Under MS-DOS 0xffff8 is used for EOF and Atari employs 0xffff by default, but both systems recognize all values from 0xffff8–0xffff as end-of-file. MS-DOS uses only 0xffff7 for bad clusters, where on Atari values 0xffff0–0xffff7 are for this

purpose (but the standard value is still 0xffff7).

OPTIONS

-a Automatically repair the filesystem. No user intervention is necessary. Whenever there is more than one method to solve a problem, the least destructive approach is used.

-A Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.

-b Make read-only boot sector check.

-c PAGE

Use DOS codepage *PAGE* to decode short file names. By default codepage 850 is used.

-d PATH

Delete the specified file. If more than one file with that name exist, the first one is deleted. This option can be given more than once.

-f Salvage unused cluster chains to files. By default, unused clusters are added to the free disk space except in auto mode (**-a**).

-F NUM

Specify FAT table *NUM* for filesystem access. By default value 0 is assumed and then the first uncorrupted FAT table is chosen. Uncorrupted means that FAT table has valid first cluster. If default value 0 is used and all FAT tables are corrupted then **fsck.fat** gives up and does not try to repair FAT filesystem. If non-zero *NUM* value is specified then **fsck.fat** uses FAT table *NUM* for repairing FAT filesystem. If FAT table *NUM* has corrupted first cluster then **fsck.fat** will repair it. In any case, if FAT filesystem has more FAT tables then repaired content of chosen FAT table is copied to other FAT tables. To repair corrupted first cluster it is required to call **fsck.fat** with non-zero *NUM* value.

-l List path names of files being processed.

-n No-operation mode: non-interactively check for errors, but don't write anything to the filesystem.

-p Same as **-a**, for compatibility with other *fsck.

-r Interactively repair the filesystem. The user is asked for advice whenever there is more than one approach to fix an inconsistency. This is the default mode and the option is only retained for backwards compatibility.

-S Consider short (8.3) file names with spaces in the middle to be invalid, like previous versions of this program did. While such file names are not forbidden by the FAT specification, and were never treated as errors by Microsoft file system checking tools, many DOS programs are unable to handle files with such names. Using this option can make them accessible to these programs.

Short file names which *start* with a space are considered invalid regardless of this option's setting.

Previous versions of this program exceptionally treated *EA DATA. SF* and *WP ROOT. SF* as valid short names; using this option does not preserve that exception.

-t Mark unreadable clusters as bad.

-u PATH

Try to undelete the specified file. **fsck.fat** tries to allocate a chain of contiguous unallocated clusters beginning with the start cluster of the undeleted file. This option can be given more than once.

-U Consider lowercase volume and boot label as invalid and allow only uppercase characters. Such labels are forbidden by the FAT specification, but they are widely used by Linux tools. Moreover MS-DOS and Windows systems do not have problems to read them. Therefore volume and boot labels with lowercase characters are by default permitted.

-v Verbose mode. Generates slightly more output.

-V Perform a verification pass. The filesystem check is repeated after the first run. The second pass should never report any fixable errors. It may take considerably longer than the first pass, because the first pass may have generated long list of modifications that have to be scanned for each disk read.

--variant *TYPE*

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

-w Write changes to disk immediately.

-y Same as **-a** (automatically repair filesystem) for compatibility with other fsck tools.

--help

Display help message describing usage and options then exit.

EXIT STATUS

0 No recoverable errors have been detected.

1 Recoverable errors have been detected or **fsck.fat** has discovered an internal inconsistency.

2 Usage error. **fsck.fat** did not access the filesystem.

FILES

fsck0000.rec, *fsck0001.rec*, ...

When recovering from a corrupted filesystem, **fsck.fat** dumps recovered data into files named *fsckNNNN.rec* in the top level directory of the filesystem.

BUGS

- Does not remove entirely empty directories.
- Should give more diagnostic messages.
- Undeleting files should use a more sophisticated algorithm.

SEE ALSO

fatlabel(8), **mkfs.fat(8)**

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

fsck.fat – check and repair MS-DOS FAT filesystems

SYNOPSIS

fsck.fat [*OPTIONS*] *DEVICE*

DESCRIPTION

fsck.fat verifies the consistency of MS-DOS filesystems and optionally tries to repair them.

The following filesystem problems can be corrected (in this order):

- FAT contains invalid cluster numbers. Cluster is changed to EOF.
- File's cluster chain contains a loop. The loop is broken.
- Bad clusters (read errors). The clusters are marked bad and they are removed from files owning them. This check is optional.
- Directories with a large number of bad entries (probably corrupt). The directory can be deleted.
- Files . and .. are non-directories. They can be deleted or renamed.
- Directories . and .. in root directory. They are deleted.
- Bad filenames. They can be renamed.
- Duplicate directory entries. They can be deleted or renamed.
- Directories with non-zero size field. Size is set to zero.
- Directory . does not point to parent directory. The start pointer is adjusted.
- Directory .. does not point to parent of parent directory. The start pointer is adjusted.
- . and .. are not the two first entries in a non-root directory. The entries are created, moving occupied slots if necessary.
- Start cluster number of a file is invalid. The file is truncated.
- File contains bad or free clusters. The file is truncated.
- File's cluster chain is longer than indicated by the size fields. The file is truncated.
- Two or more files share the same cluster(s). All but one of the files are truncated. If the file being truncated is a directory file that has already been read, the filesystem check is restarted after truncation.
- File's cluster chain is shorter than indicated by the size fields. The file is truncated.
- Volume label in root directory or label in boot sector is invalid. Invalid labels are removed.
- Volume label in root directory and label in boot sector are different. Volume label from root directory is copied to boot sector.
- Clusters are marked as used but are not owned by a file. They are marked as free.

Additionally, the following problems are detected, but not repaired:

- Invalid parameters in boot sector

When **fsck.fat** checks a filesystem, it accumulates all changes in memory and performs them only after all checks are complete. This can be disabled with the **-w** option.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

There are some minor differences in Atari format: Some boot sector fields are interpreted slightly different, and the special FAT entries for end-of-file and bad cluster can be different. Under MS-DOS 0xffff8 is used for EOF and Atari employs 0xffff by default, but both systems recognize all values from 0xffff8–0xffff as end-of-file. MS-DOS uses only 0xffff7 for bad clusters, where on Atari values 0xffff0–0xffff7 are for this

purpose (but the standard value is still 0xffff7).

OPTIONS

-a Automatically repair the filesystem. No user intervention is necessary. Whenever there is more than one method to solve a problem, the least destructive approach is used.

-A Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.

-b Make read-only boot sector check.

-c PAGE

Use DOS codepage *PAGE* to decode short file names. By default codepage 850 is used.

-d PATH

Delete the specified file. If more than one file with that name exist, the first one is deleted. This option can be given more than once.

-f Salvage unused cluster chains to files. By default, unused clusters are added to the free disk space except in auto mode (**-a**).

-F NUM

Specify FAT table *NUM* for filesystem access. By default value 0 is assumed and then the first uncorrupted FAT table is chosen. Uncorrupted means that FAT table has valid first cluster. If default value 0 is used and all FAT tables are corrupted then **fsck.fat** gives up and does not try to repair FAT filesystem. If non-zero *NUM* value is specified then **fsck.fat** uses FAT table *NUM* for repairing FAT filesystem. If FAT table *NUM* has corrupted first cluster then **fsck.fat** will repair it. In any case, if FAT filesystem has more FAT tables then repaired content of chosen FAT table is copied to other FAT tables. To repair corrupted first cluster it is required to call **fsck.fat** with non-zero *NUM* value.

-l List path names of files being processed.

-n No-operation mode: non-interactively check for errors, but don't write anything to the filesystem.

-p Same as **-a**, for compatibility with other *fsck.

-r Interactively repair the filesystem. The user is asked for advice whenever there is more than one approach to fix an inconsistency. This is the default mode and the option is only retained for backwards compatibility.

-S Consider short (8.3) file names with spaces in the middle to be invalid, like previous versions of this program did. While such file names are not forbidden by the FAT specification, and were never treated as errors by Microsoft file system checking tools, many DOS programs are unable to handle files with such names. Using this option can make them accessible to these programs.

Short file names which *start* with a space are considered invalid regardless of this option's setting.

Previous versions of this program exceptionally treated *EA DATA. SF* and *WP ROOT. SF* as valid short names; using this option does not preserve that exception.

-t Mark unreadable clusters as bad.

-u PATH

Try to undelete the specified file. **fsck.fat** tries to allocate a chain of contiguous unallocated clusters beginning with the start cluster of the undeleted file. This option can be given more than once.

-U Consider lowercase volume and boot label as invalid and allow only uppercase characters. Such labels are forbidden by the FAT specification, but they are widely used by Linux tools. Moreover MS-DOS and Windows systems do not have problems to read them. Therefore volume and boot labels with lowercase characters are by default permitted.

-v Verbose mode. Generates slightly more output.

-V Perform a verification pass. The filesystem check is repeated after the first run. The second pass should never report any fixable errors. It may take considerably longer than the first pass, because the first pass may have generated long list of modifications that have to be scanned for each disk read.

--variant *TYPE*

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

-w Write changes to disk immediately.

-y Same as **-a** (automatically repair filesystem) for compatibility with other fsck tools.

--help

Display help message describing usage and options then exit.

EXIT STATUS

0 No recoverable errors have been detected.

1 Recoverable errors have been detected or **fsck.fat** has discovered an internal inconsistency.

2 Usage error. **fsck.fat** did not access the filesystem.

FILES

fsck0000.rec, *fsck0001.rec*, ...

When recovering from a corrupted filesystem, **fsck.fat** dumps recovered data into files named *fsckNNNN.rec* in the top level directory of the filesystem.

BUGS

- Does not remove entirely empty directories.
- Should give more diagnostic messages.
- Undeleting files should use a more sophisticated algorithm.

SEE ALSO

fatlabel(8), **mkfs.fat(8)**

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

fsck.fat – check and repair MS-DOS FAT filesystems

SYNOPSIS

fsck.fat [*OPTIONS*] *DEVICE*

DESCRIPTION

fsck.fat verifies the consistency of MS-DOS filesystems and optionally tries to repair them.

The following filesystem problems can be corrected (in this order):

- FAT contains invalid cluster numbers. Cluster is changed to EOF.
- File's cluster chain contains a loop. The loop is broken.
- Bad clusters (read errors). The clusters are marked bad and they are removed from files owning them. This check is optional.
- Directories with a large number of bad entries (probably corrupt). The directory can be deleted.
- Files . and .. are non-directories. They can be deleted or renamed.
- Directories . and .. in root directory. They are deleted.
- Bad filenames. They can be renamed.
- Duplicate directory entries. They can be deleted or renamed.
- Directories with non-zero size field. Size is set to zero.
- Directory . does not point to parent directory. The start pointer is adjusted.
- Directory .. does not point to parent of parent directory. The start pointer is adjusted.
- . and .. are not the two first entries in a non-root directory. The entries are created, moving occupied slots if necessary.
- Start cluster number of a file is invalid. The file is truncated.
- File contains bad or free clusters. The file is truncated.
- File's cluster chain is longer than indicated by the size fields. The file is truncated.
- Two or more files share the same cluster(s). All but one of the files are truncated. If the file being truncated is a directory file that has already been read, the filesystem check is restarted after truncation.
- File's cluster chain is shorter than indicated by the size fields. The file is truncated.
- Volume label in root directory or label in boot sector is invalid. Invalid labels are removed.
- Volume label in root directory and label in boot sector are different. Volume label from root directory is copied to boot sector.
- Clusters are marked as used but are not owned by a file. They are marked as free.

Additionally, the following problems are detected, but not repaired:

- Invalid parameters in boot sector

When **fsck.fat** checks a filesystem, it accumulates all changes in memory and performs them only after all checks are complete. This can be disabled with the **-w** option.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

There are some minor differences in Atari format: Some boot sector fields are interpreted slightly different, and the special FAT entries for end-of-file and bad cluster can be different. Under MS-DOS 0xffff8 is used for EOF and Atari employs 0xffff by default, but both systems recognize all values from 0xffff8–0xffff as end-of-file. MS-DOS uses only 0xffff7 for bad clusters, where on Atari values 0xffff0–0xffff7 are for this

purpose (but the standard value is still 0xffff7).

OPTIONS

-a Automatically repair the filesystem. No user intervention is necessary. Whenever there is more than one method to solve a problem, the least destructive approach is used.

-A Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.

-b Make read-only boot sector check.

-c PAGE

Use DOS codepage *PAGE* to decode short file names. By default codepage 850 is used.

-d PATH

Delete the specified file. If more than one file with that name exist, the first one is deleted. This option can be given more than once.

-f Salvage unused cluster chains to files. By default, unused clusters are added to the free disk space except in auto mode (**-a**).

-F NUM

Specify FAT table *NUM* for filesystem access. By default value 0 is assumed and then the first uncorrupted FAT table is chosen. Uncorrupted means that FAT table has valid first cluster. If default value 0 is used and all FAT tables are corrupted then **fsck.fat** gives up and does not try to repair FAT filesystem. If non-zero *NUM* value is specified then **fsck.fat** uses FAT table *NUM* for repairing FAT filesystem. If FAT table *NUM* has corrupted first cluster then **fsck.fat** will repair it. In any case, if FAT filesystem has more FAT tables then repaired content of chosen FAT table is copied to other FAT tables. To repair corrupted first cluster it is required to call **fsck.fat** with non-zero *NUM* value.

-l List path names of files being processed.

-n No-operation mode: non-interactively check for errors, but don't write anything to the filesystem.

-p Same as **-a**, for compatibility with other *fsck.

-r Interactively repair the filesystem. The user is asked for advice whenever there is more than one approach to fix an inconsistency. This is the default mode and the option is only retained for backwards compatibility.

-S Consider short (8.3) file names with spaces in the middle to be invalid, like previous versions of this program did. While such file names are not forbidden by the FAT specification, and were never treated as errors by Microsoft file system checking tools, many DOS programs are unable to handle files with such names. Using this option can make them accessible to these programs.

Short file names which *start* with a space are considered invalid regardless of this option's setting.

Previous versions of this program exceptionally treated *EA DATA. SF* and *WP ROOT. SF* as valid short names; using this option does not preserve that exception.

-t Mark unreadable clusters as bad.

-u PATH

Try to undelete the specified file. **fsck.fat** tries to allocate a chain of contiguous unallocated clusters beginning with the start cluster of the undeleted file. This option can be given more than once.

-U Consider lowercase volume and boot label as invalid and allow only uppercase characters. Such labels are forbidden by the FAT specification, but they are widely used by Linux tools. Moreover MS-DOS and Windows systems do not have problems to read them. Therefore volume and boot labels with lowercase characters are by default permitted.

-v Verbose mode. Generates slightly more output.

-V Perform a verification pass. The filesystem check is repeated after the first run. The second pass should never report any fixable errors. It may take considerably longer than the first pass, because the first pass may have generated long list of modifications that have to be scanned for each disk read.

--variant *TYPE*

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

-w Write changes to disk immediately.

-y Same as **-a** (automatically repair filesystem) for compatibility with other fsck tools.

--help

Display help message describing usage and options then exit.

EXIT STATUS

0 No recoverable errors have been detected.

1 Recoverable errors have been detected or **fsck.fat** has discovered an internal inconsistency.

2 Usage error. **fsck.fat** did not access the filesystem.

FILES

fsck0000.rec, *fsck0001.rec*, ...

When recovering from a corrupted filesystem, **fsck.fat** dumps recovered data into files named *fsckNNNN.rec* in the top level directory of the filesystem.

BUGS

- Does not remove entirely empty directories.
- Should give more diagnostic messages.
- Undeleting files should use a more sophisticated algorithm.

SEE ALSO

fatlabel(8), **mkfs.fat(8)**

HOMEPAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

fsck.xfs – do nothing, successfully

SYNOPSIS

fsck.xfs [*filesys* ...]

DESCRIPTION

fsck.xfs is called by the generic Linux **fsck(8)** program at startup to check and repair an XFS filesystem. XFS is a journaling filesystem and performs recovery at **mount(8)** time if necessary, so **fsck.xfs** simply exits with a zero exit status.

If you wish to check the consistency of an XFS filesystem, or repair a damaged or corrupt XFS filesystem, see **xfs_repair(8)**.

However, the system administrator can force **fsck.xfs** to run **xfs_repair(8)** at boot time by creating a */forcefsck* file or booting the system with "fsck.mode=force" on the kernel command line.

FILES

/etc/fstab.

SEE ALSO

fsck(8), **fstab(5)**, **xfs(5)**, **xfs_repair(8)**.

NAME

fstab-decode – run a command with fstab-encoded arguments

SYNOPSIS

fstab-decode *COMMAND* [*ARGUMENT* ...]

DESCRIPTION

fstab-decode decodes escapes (such as newline characters and other whitespace) in the specified *ARGUMENT*'s and uses them to run *COMMAND*. The argument escaping uses the same rules as path escaping in */etc/fstab*, */etc/mtab* and */proc/mtab*.

In essence **fstab-decode** can be used anytime we want to pass multiple parameters to a command as a list of command line arguments. It turns output like this:

```
/root  
/mnt/remote-disk  
/home
```

Into one long list of parameters, "/root /mnt/remote-disk /home". This can be useful when trying to work with multiple filesystems at once. For instance, we can use it to umount multiple NFS shares. This program also removes whitespace and other characters which might cause programs such as mount or umount to fail.

EXIT STATUS

fstab-decode exits with status 127 if *COMMAND* can't be run. Otherwise it exits with the status returned by *COMMAND*.

EXAMPLES

The following example reads fstab, finds all instances of VFAT filesystems and prints their mount points (argument 2 in the fstab file). **fstab-decode** then runs the specified program, **umount**, and passes it the list of VFAT mountpoints. This unmounts all VFAT partitions.

```
fstab-decode umount $(awk '$3 == "vfat" { print $2 }' /etc/fstab)
```

SEE ALSO

fstab(5)

NAME

fstab – static information about the filesystems

SYNOPSIS

/etc/fstab

DESCRIPTION

The file **fstab** contains descriptive information about the filesystems the system can mount. **fstab** is only read by programs, and not written; it is the duty of the system administrator to properly create and maintain this file. The order of records in **fstab** is important because **fsck(8)**, **mount(8)**, and **umount(8)** sequentially iterate through **fstab** doing their thing.

Each filesystem is described on a separate line. Fields on each line are separated by tabs or spaces. Lines starting with '#' are comments. Blank lines are ignored.

The following is a typical example of an **fstab** entry:

```
LABEL=t-home2      /home        ext4      defaults,auto_da_alloc      0      2
```

The first field (*fs_spec*).

This field describes the block special device, remote filesystem or filesystem image for loop device to be mounted or swap file or swap partition to be enabled.

For ordinary mounts, it will hold (a link to) a block special device node (as created by **mknod(2)**) for the device to be mounted, like */dev/cdrom* or */dev/sdb7*. For NFS mounts, this field is *<host>:<dir>*, e.g., *knuth.aeb.nl:/*. For filesystems with no storage, any string can be used, and will show up in **df(1)** output, for example. Typical usage is *proc* for **procfs**; *mem*, *none*, or *tmpfs* for **tmpfs**. Other special filesystems, like **udev** and **sysfs**, are typically not listed in **fstab**.

LABEL=<label> or **UUID=<uuid>** may be given instead of a device name. This is the recommended method, as device names are often a coincidence of hardware detection order, and can change when other disks are added or removed. For example, 'LABEL=Boot' or '**UUID=3e6be9de-8139-11d1-9106-a43f08d823a6**'. (Use a filesystem-specific tool like **e2label(8)**, **xfs_admin(8)**, or **fatlabel(8)** to set LABELs on filesystems).

It's also possible to use **PARTUUID=** and **PARTLABEL=**. These partitions identifiers are supported for example for GUID Partition Table (GPT).

See **mount(8)**, **blkid(8)** or **lsblk(8)** for more details about device identifiers.

Note that **mount(8)** uses UUIDs as strings. The string representation of the UUID should be based on lower case characters. But when specifying the volume ID of FAT or NTFS file systems upper case characters are used (e.g **UUID="A40D-85E7"** or **UUID="61DB7756DB7779B3"**).

The second field (*fs_file*).

This field describes the mount point (target) for the filesystem. For swap partitions, this field should be specified as 'none'. If the name of the mount point contains spaces or tabs these can be escaped as '\040' and '\011' respectively.

The third field (*fs_vfstype*).

This field describes the type of the filesystem. Linux supports many filesystem types: ext4, xfs, btrfs, f2fs, vfat, ntfs, hfsplus, tmpfs, sysfs, proc, iso9660, udf, squashfs, nfs, cifs, and many more. For more details, see **mount(8)**.

An entry *swap* denotes a file or partition to be used for swapping, cf. **swapon(8)**. An entry *none* is useful for bind or move mounts.

More than one type may be specified in a comma-separated list.

mount(8) and **umount(8)** support filesystem *subtypes*. The subtype is defined by '.subtype' suffix. For example 'fuse.sshfs'. It's recommended to use subtype notation rather than add any prefix to the first fstab field (for example 'sshfs#example.com' is deprecated).

The fourth field (*fs_mntops*).

This field describes the mount options associated with the filesystem.

It is formatted as a comma-separated list of options. It contains at least the type of mount (**ro** or **rw**), plus any additional options appropriate to the filesystem type (including performance-tuning options). For details, see **mount(8)** or **swapon(8)**.

Basic filesystem-independent options are:

defaults

use default options: rw, suid, dev, exec, auto, nouser, and async.

noauto

do not mount when **mount -a** is given (e.g., at boot time)

user

allow a user to mount

owner

allow device owner to mount

comment

or x-<name> for use by fstab-maintaining programs

nofail

do not report errors for this device if it does not exist.

The fifth field (*fs_freq*).

This field is used by **dump(8)** to determine which filesystems need to be dumped. Defaults to zero (don't dump) if not present.

The sixth field (*fs_passno*).

This field is used by **fsck(8)** to determine the order in which filesystem checks are done at boot time. The root filesystem should be specified with a *fs_passno* of 1. Other filesystems should have a *fs_passno* of 2. Filesystems within a drive will be checked sequentially, but filesystems on different drives will be checked at the same time to utilize parallelism available in the hardware. Defaults to zero (don't check the filesystem) if not present.

FILES

/etc/fstab, <fstab.h>

NOTES

The proper way to read records from **fstab** is to use the routines **getmntent(3)** or **libmount**.

The keyword **ignore** as a filesystem type (3rd field) is no longer supported by the pure libmount based mount utility (since util-linux v2.22).

HISTORY

The ancestor of this **fstab** file format appeared in 4.0BSD.

SEE ALSO

getmntent(3), **fs(5)**, **findmnt(8)**, **mount(8)**, **swapon(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

fstab is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

tnftp — Internet file transfer program

SYNOPSIS

```
tnftp [ -46AadefginpRtVv? ] [-N netrc] [-o output] [-P port] [-q quittime]
      [ -r retry ] [ -s srcaddr ] [ -T dir,max[,inc] ] [ -x xfersize ]
      [[user@]host [port]] [[user@]host:[path][/]][file://path]
      [ftp://[user[:password]@]host[:port]/path[/][;type=type]]
      [http://[user[:password]@]host[:port]/path]
      [https://[user[:password]@]host[:port]/path] ...
tnftp -u url file ...
```

DESCRIPTION

tnftp is the user interface to the Internet standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

The last five arguments will fetch a file using the FTP or HTTP protocols, or by direct copying, into the current directory. This is ideal for scripts. Refer to **AUTO-FETCHING FILES** below for more information.

Options may be specified at the command line, or to the command interpreter.

- 4 Forces **tnftp** to only use IPv4 addresses.
- 6 Forces **tnftp** to only use IPv6 addresses.
- A Force active mode FTP. By default, **tnftp** will try to use passive mode FTP and fall back to active mode if passive is not supported by the server. This option causes **tnftp** to always use an active connection. It is only useful for connecting to very old servers that do not implement passive mode properly.
- a Causes **tnftp** to bypass normal login procedure, and use an anonymous login instead.
- d Enables debugging.
- e Disables command line editing. This is useful for Emacs ange-ftp mode.
- f Forces a cache reload for transfers that go through the FTP or HTTP proxies.
- g Disables file name globbing.
- i Turns off interactive prompting during multiple file transfers.
- N *netrc* Use *netrc* instead of *~/.netrc*. Refer to **THE .NETRC FILE** for more information.
- n Restraints **tnftp** from attempting “auto-login” upon initial connection for non auto-fetch transfers. If auto-login is enabled, **tnftp** will check the *.netrc* (see below) file in the user’s home directory for an entry describing an account on the remote machine. If no entry exists, **tnftp** will prompt for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompt for a password and an account with which to login. To override the auto-login for auto-fetch transfers, specify the username (and optionally, password) as appropriate.
- o *output* When auto-fetching files, save the contents in *output*. *output* is parsed according to the **FILE NAMING CONVENTIONS** below. If *output* is not ‘-’ or doesn’t start with ‘|’, then only the first file specified will be retrieved into *output*; all other files will be retrieved into the basename of their remote name.

- P port** Sets the port number to *port*.
- p** Enable passive mode operation for use behind connection filtering firewalls. This option has been deprecated as **tnftp** now tries to use passive mode by default, falling back to active mode if the server does not support passive connections.
- q *quittime*** Quit if the connection has stalled for *quittime* seconds.
- R** Restart all non-proxied auto-fetches.
- r *wait*** Retry the connection attempt if it failed, pausing for *wait* seconds.
- s *srcaddr*** Uses *srcaddr* as the local IP address for all connections.
- t** Enables packet tracing.
- T *direction,maximum[,increment]*** Set the maximum transfer rate for *direction* to *maximum* bytes/second, and if specified, the increment to *increment* bytes/second. Refer to **rate** for more information.
- u *url file ...*** Upload files on the command line to *url* where *url* is one of the ‘**ftp://**’ URL types as supported by auto-fetch (with an optional target filename for single file uploads), and *file* is one or more local files to be uploaded.
- v** Disable **verbose** and **progress**, overriding the default of enabled when output is to a terminal.
- vv** Enable **verbose** and **progress**. This is the default if output is to a terminal (and in the case of **progress**, **tnftp** is the foreground process). Forces **tnftp** to show all responses from the remote server, as well as report on data transfer statistics.
- x *xfersize*** Set the size of the socket send and receive buffers to *xfersize*. Refer to **xferbuf** for more information.
- ?** Display help to stdout, and exit.

The client host with which **tnftp** is to communicate may be specified on the command line. If this is done, **tnftp** will immediately attempt to establish a connection to an FTP server on that host; otherwise, **tnftp** will enter its command interpreter and await instructions from the user. When **tnftp** is a waiting commands from the user the prompt **f tp>** is provided to the user. The following commands are recognized by **tnftp**:

- ! [command [args]]**
Invoke an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.
- \$ macro-name [args]**
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.
- account [passwd]**
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]

Append a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

ascii Set the file transfer **type** to network ASCII. This is the default type.**bell** Arrange that a bell be sounded after each file transfer command is completed.**binary** Set the file transfer **type** to support binary image transfer.**bye** Terminate the FTP session with the remote server and exit **tnftp**. An end of file will also terminate the session and exit.**case** Toggle remote computer file name case mapping during **get**, **mget** and **mput** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.**cd** *remote-directory*

Change the working directory on the remote machine to *remote-directory*.

cdup Change the remote machine working directory to the parent of the current remote machine working directory.**chmod** *mode* *remote-file*

Change the permission modes of the file *remote-file* on the remote system to *mode*.

close Terminate the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.**cr** Toggle carriage return stripping during ascii type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single linefeed record delimiter. Records on non-UNIX remote systems may contain single linefeeds; when an ascii type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.**debug** [*debug-value*]

Toggle debugging mode. If an optional *debug-value* is specified it is used to set the debugging level. When debugging is on, **tnftp** prints each command sent to the remote machine, preceded by the string -->.

delete *remote-file*

Delete the file *remote-file* on the remote machine.

dir [*remote-path* [*local-file*]]

Print a listing of the contents of a directory on the remote machine. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems will produce output from the command **ls -l**. If *remote-path* is left unspecified, the current working directory is used. If interactive prompting is on, **tnftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **dir** output. If no local file is specified, or if *local-file* is ‘-’, the output is sent to the terminal.

disconnect

A synonym for **close**.

edit Toggle command line editing, and context sensitive command and file completion. This is automatically enabled if input is from a terminal, and disabled otherwise.

epsv, epsv4, epsv6

Toggle the use of the extended EPSV and EPRT commands on all IP, IPv4, and IPv6 connections respectively. First tryEPSV/EPRT, and then PASV/PORT. This is enabled by default. If an extended command fails then this option will be temporarily disabled for the duration of the current connection, or until **epsv**, **epsv4**, or **epsv6** is executed again.

exit A synonym for **bye**.

features Display what features the remote server supports (using the FEAT command).

fget *localfile*

Retrieve the files listed in *localfile*, which has one line per filename.

form *format*

Set the file transfer **form** to *format*. The default (and only supported) format is “non-print”.

ftp *host* [*port*]

A synonym for **open**.

gate [*host* [*port*]]

Toggle gate-ftp mode, which used to connect through the TIS FWTK and Gauntlet FTP proxies. This will not be permitted if the gate-ftp server hasn't been set (either explicitly by the user, or from the FTPSERVER environment variable). If*host* is given, then gate-ftp mode will be enabled, and the gate-ftp server will be set to *host*. If*port* is also given, that will be used as the port to connect to on the gate-ftp server.

get *remote-file* [*local-file*]

Retrieve the *remote-file* and store it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type**, **form**, **mode**, and **structure** are used while transferring the file.

glob

Toggle filename expansion for **mdelete**, **mget**, **mput**, and **mreget**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in csh(1). For **mdelete**, **mget**, and **mreget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file: the exact result depends on the foreign operating system and FTP server, and can be previewed by doing ‘**mls** *remote-files* –’. Note:**mget**, **mput** and **mreget** are not meant to transfer entire directory subtrees of files. That can be done by transferring a **tar**(1) archive of the subtree (in binary mode).

hash [*size*]

Toggle hash-sign (#) printing for each data block transferred. The size of a data block defaults to 1024 bytes. This can be changed by specifying *size* in bytes. Enabling **hash** disables **progress**.

help [*command*]

Print an informative message about the meaning of *command*. If no argument is given, **tnftp** prints a list of the known commands.

idle [*seconds*]

Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, the current inactivity timer is printed.

image A synonym for **binary**.

lcd [directory]

Change the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

less file

A synonym for **page**.

lpage local-file

Display *local-file* with the program specified by the **set pager** option.

lpwd

Print the working directory on the local machine.

ls [remote-path [local-file]]

A synonym for **dir**.

macdef macro-name

Define a macro. Subsequent lines are stored as the macro *macro-name*; a null line (consecutive newline characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macro names can be a maximum of 8 characters. Macros are only applicable to the current session they are defined within (or if defined outside a session, to the session invoked with the next **open** command), and remain defined until a **close** command is executed. To invoke a macro, use the **\$** command (see above).

The macro processor interprets '\$' and '\' as special characters. A '\$' followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A '\$' followed by an 'i' signals the macro processor that the executing macro is to be looped. On the first pass '\$i' is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A '\' followed by any character is replaced by that character. Use the '\' to prevent special treatment of the '\$'.

mdelete [remote-files]

Delete the *remote-files* on the remote machine.

mdir remote-files local-file

Like **dir**, except multiple remote files may be specified. If interactive prompting is on, **tnftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mdir** output.

mget remote-files

Expand the *remote-files* on the remote machine and do a **get** for each file name thus produced. See **glob** for details on the filename expansion. Resulting file names will then be processed according to **case**, **ntrans**, and **nmap** settings. Files are transferred into the local working directory, which can be changed with **lcd directory**; new local directories can be created with '! **mkdir directory**'.

mkdir directory-name

Make a directory on the remote machine.

mls remote-files local-file

Like **ls**, except multiple remote files may be specified, and the *local-file* must be specified. If interactive prompting is on, **tnftp** will prompt the user to verify that the last argument is indeed the target local file for receiving **mls** output.

mlsd [remote-path]

Display the contents of *remote-path* (which should default to the current directory if not given) in a machine-parsable form, using MLSD. The format of display can be changed with

‘remopts mlst ...’.

mlst [*remote-path*]
 Display the details about *remote-path* (which should default to the current directory if not given) in a machine-parsable form, using MLST. The format of display can be changed with ‘remopts mlst ...’.

mode *mode-name*
 Set the file transfer **mode** to *mode-name*. The default (and only supported) mode is “stream”.

modtime *remote-file*
 Show the last modification time of the file on the remote machine, in RFC 2822 format.

more *file*
 A synonym for **page**.

mput *local-files*
 Expand wild cards in the list of local files given as arguments and do a **put** for each file in the resulting list. See **glob** for details of filename expansion. Resulting file names will then be processed according to **ntrans** and **nmap** settings.

mreget *remote-files*
 As per **mget**, but performs a **reget** instead of **get**.

msend *local-files*
 A synonym for **mput**.

newer *remote-file* [*local-file*]
 Get the file only if the modification time of the remote file is more recent than the file on the current system. If the file does not exist on the current system, the remote file is considered **newer**. Otherwise, this command is identical to **get**.

nlist [*remote-path* [*local-file*]]
 A synonym for **ls**.

nmap [*inpattern* *outpattern*]
 Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non- UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*.

inpattern is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences ‘\$1’, ‘\$2’, ..., ‘\$9’ in *inpattern*. Use ‘\’ to prevent this special treatment of the ‘\$’ character. All other characters are treated literally, and are used to determine the **nmap** [*inpattern*] variable values. For example, given *inpattern* ‘\$1.\$2’ and the remote file name ‘mydata.data’, ‘\$1’ would have the value ‘mydata’, and ‘\$2’ would have the value ‘data’.

The *outpattern* determines the resulting mapped filename. The sequences ‘\$1’, ‘\$2’, ..., ‘\$9’ are replaced by any value resulting from the *inpattern* template. The sequence ‘\$0’ is replaced by the original filename. Additionally, the sequence “[*seq1*, *seq2*]” is replaced by *seq1* if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command

```
nmap $1.$2.$3 [${1,$2}].[$2,file]
```

would yield the output filename ‘myfile.data’ for input filenames ‘myfile.data’ and ‘myfile.data.old’, ‘myfile.file’ for the input filename ‘myfile’, and ‘myfile myfile’ for the input filename ‘.myfile’. Spaces may be included in *outpattern*, as in the example:

```
nmap $1 sed s/*$// > $1
```

Use the ‘\’ character to prevent special treatment of the ‘\$’, ‘[’, ‘]’, and ‘,’ characters.

ntrans [*inchars* [*outchars*]]

Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non- UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character’s position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *host* [*port*]

Establish a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, **tnftp** will attempt to contact an FTP server at that port. If the **set auto-login** option is on (default), **tnftp** will also attempt to automatically log the user in to the FTP server (see below).

page *file*

Retrieve *file* and display with the program specified by the **set pager** option.

passive [*auto*]

Toggle passive mode (if no arguments are given). If *auto* is given, act as if **FTPMODE** is set to ‘auto’. If passive mode is turned on (default), **tnftp** will send a PASV command for all data connections instead of a PORT command. The PASV command requests that the remote server open a port for the data connection and return the address of that port. The remote server listens on that port and the client connects to it. When using the more traditional PORT command, the client listens on a port and sends that address to the remote server, who connects back to it. Passive mode is useful when using **tnftp** through a gateway router or host that controls the directionality of traffic. (Note that though FTP servers are required to support the PASV command by RFC 1123, some do not.)

pdir [*remote-path*]

Perform **dir** [*remote-path*], and display the result with the program specified by the **set pager** option.

pls [*remote-path*]

Perform **ls** [*remote-path*], and display the result with the program specified by the **set pager** option.

pmlsd [*remote-path*]

Perform **mlsd** [*remote-path*], and display the result with the program specified by the **set pager** option.

preserve Toggle preservation of modification times on retrieved files.

progress Toggle display of transfer progress bar. The progress bar will be disabled for a transfer that has *local-file* as ‘-’ or a command that starts with ‘|’. Refer to FILE N AMING CONVENTIONS for more information. Enabling **progress** disables **hash**.

prompt Toggle interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** will transfer all files, and any **mdelete** will delete all files.

When prompting is on, the following commands are available at a prompt:

- a** Answer ‘yes’ to the current file, and automatically answer ‘yes’ to any remaining files for the current command.
- n** Answer ‘no’, and do not transfer the file.
- p** Answer ‘yes’ to the current file, and turn off prompt mode (as is “prompt off” had been given).
- q** Terminate the current operation.
- y** Answer ‘yes’, and transfer the file.
- ?** Display a help message.

Any other response will answer ‘yes’ to the current file.

proxy *ftp-command*

Execute an FTP command on a secondary control connection. This command allows simultaneous connection to two remote FTP servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command ‘proxy ?’ to see other FTP commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** will not define new macros during the auto-login process, **close** will not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the FTP protocol PASV command by the server on the secondary control connection.

put *local-file* [*remote-file*]

Store a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any **ntrans** or **nmap** settings in naming the remote file. File transfer uses the current settings for **type**, **format**, **mode**, and **structure**.

pwd

Print the name of the current working directory on the remote machine.

quit

A synonym for **bye**.

quote [*arg* . . .]

The arguments specified are sent, verbatim, to the remote FTP server.

rate *direction* [*maximum* [*increment*]]

Throttle the maximum transfer rate to *maximum* bytes/second. If *maximum* is 0, disable the throttle.

direction may be one of:

all Both directions.
get Incoming transfers.
put Outgoing transfers.

maximum can be modified on the fly by *increment* bytes (default: 1024) each time a given signal is received:

SIGUSR1 Increment *maximum* by *increment* bytes.

SIGUSR2 Decrement *maximum* by *increment* bytes. The result must be a positive number.

If *maximum* is not supplied, the current throttle rates are displayed.

Note: **rate** is not yet implemented for ascii mode transfers.

rcvbuf *size*

Set the size of the socket receive buffer to *size*.

recv *remote-file* [*local-file*]

A synonym for **get**.

reget *remote-file* [*local-file*]

reget acts like **get**, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that are prone to dropping connections.

remopts *command* [*command-options*]

Set options on the remote FTP server for *command* to *command-options* (whose absence is handled on a command-specific basis). Remote FTP commands known to support options include: MLST (used for MLSD and MLST).

rename [*from* [*to*]]

Rename the file *from* on the remote machine, to the file *to*.

reset

Clear reply queue. This command re-synchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

restart *marker*

Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, *marker* is usually a byte offset into the file.

rhelp [*command-name*]

Request help from the remote FTP server. If a *command-name* is specified it is supplied to the server as well.

rmdir *directory-name*

Delete a directory on the remote machine.

rstatus [*remote-file*]

With no arguments, show status of remote machine. If *remote-file* is specified, show status of *remote-file* on remote machine.

runique

Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a ‘.1’ is appended to the name. If the resulting name matches another existing file, a ‘.2’ is appended to the original name. If this process continues up to .99, an error message is printed, and the transfer does not take place. The generated unique filename will be reported. Note that **runique** will not

affect local files generated from a shell command (see below). The default value is off.

send *local-file* [*remote-file*]
A synonym for **put**.

sendport Toggle the use of PORT commands. By default, **tnftp** will attempt to use a PORT command when establishing a connection for each data transfer. The use of PORT commands can prevent delays when performing multiple file transfers. If the PORT command fails, **tnftp** will use the default data port. When the use of PORT commands is disabled, no attempt will be made to use PORT commands for each data transfer. This is useful for certain FTP implementations which do ignore PORT commands but, incorrectly, indicate they've been accepted.

set [*option value*]

Set *option* to *value*. If *option* and *value* are not given, display all of the options and their values. The currently supported options are:

anonpass	Defaults to \$FTPANONPASS
ftp_proxy	Defaults to \$ftp_proxy.
http_proxy	Defaults to \$http_proxy.
https_proxy	Defaults to \$https_proxy.
no_proxy	Defaults to \$no_proxy.
pager	Defaults to \$PAGER.
prompt	Defaults to \$FTPPROMPT.
rprompt	Defaults to \$FTPRPROMPT.

site [*arg . . .*]

The arguments specified are sent, verbatim, to the remote FTP server as a SITE command.

size *remote-file*

Return size of *remote-file* on remote machine.

sndbuf *size*

Set the size of the socket send buffer to *size*.

status Show the current status of **tnftp**.

struct *struct-name*

Set the file transfer *structure* to *struct-name*. The default (and only supported) structure is "file".

sunique Toggle storing of files on remote machine under unique file names. The remote FTP server must support FTP protocol STOU command for successful completion. The remote server will report unique name. Default value is off.

system Show the type of operating system running on the remote machine.

tenex Set the file transfer type to that needed to talk to TENEX machines.

throttle A synonym for **rate**.

trace Toggle packet tracing.

type [*type-name*]

Set the file transfer **type** to *type-name*. If no type is specified, the current type is printed. The default type is network ASCII.

umask [*newmask*]

Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

unset *option*

Unset *option*. Refer to **set** for more information.

usage *command*

Print the usage message for *command*.

user *user-name* [*password* [*account*]]

Identify yourself to the remote FTP server. If the *password* is not specified and the server requires it, **tnftp** will prompt the user for it (after disabling local echo). If an *account* field is not specified, and the FTP server requires it, the user will be prompted for it. If an *account* field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless **tnftp** is invoked with “auto-login” disabled, this process is done automatically on initial connection to the FTP server.

verbose Toggle verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on.

xferbuf *size*

Set the size of the socket send and receive buffers to *size*.

? [*command*]

A synonym for **help**.

Command arguments which have embedded spaces may be quoted with quote ‘"’ marks.

Commands which toggle settings can take an explicit **on** or **off** argument to force the setting appropriately.

Commands which take a byte count as an argument (e.g., **hash**, **rate**, and **xferbuf**) support an optional suffix on the argument which changes the interpretation of the argument. Supported suffixes are:

- b Causes no modification. (Optional)
- k Kilo; multiply the argument by 1024
- m Mega; multiply the argument by 1048576
- g Giga; multiply the argument by 1073741824

If **tnftp** receives a SIGINFO (see the **status** argument of **stty(1)**) or SIGQUIT signal whilst a transfer is in progress, the current transfer rate statistics will be written to the standard error output, in the same format as the standard completion message.

AUTO-FETCHING FILES

In addition to standard commands, this version of **tnftp** supports an auto-fetch feature. To enable auto-fetch, simply pass the list of hostnames/files on the command line.

The following formats are valid syntax for an auto-fetch element:

[*user@host* : [*path*] [/]
“Classic” FTP format.

If *path* contains a glob character and globbing is enabled, (see **glob**), then the equivalent of ‘**mget path**’ is performed.

If the directory component of *path* contains no globbing characters, it is stored locally with the name basename (see **basename(1)**) of **path**, in the current directory. Otherwise, the full remote name is used as the local name, relative to the local root directory.

ftp://[user[:password]@]host[:port]/path[/][;type=type]

An FTP URL, retrieved using the FTP protocol if **set ftp_proxy** isn't defined. Otherwise, transfer the URL using HTTP via the proxy defined in **set http_proxy**. If **set ftp_proxy** isn't defined and *user* is given, login as *user*. In this case, use *password* if supplied, otherwise prompt the user for one.

If a suffix of ‘;type=A’ or ‘;type=I’ is supplied, then the transfer type will take place as ascii or binary (respectively). The default transfer type is binary.

In order to be compliant with RFC 3986, **tnftp** interprets the *path* part of an ‘**ftp://**’ auto-fetch URL as follows:

- The ‘/’ immediately after the *host[:port]* is interpreted as a separator before the *path*, and not as part of the *path* itself.
- The *path* is interpreted as a ‘/’-separated list of name components. For all but the last such component, **tnftp** performs the equivalent of a **cd** command. For the last path component, **tnftp** performs the equivalent of a **get** command.
- Empty name components, which result from ‘//’ within the *path*, or from an extra ‘/’ at the beginning of the *path*, will cause the equivalent of a **cd** command without a directory name. This is unlikely to be useful.
- Any ‘%XX’ codes (perRFC 3986) within the path components are decoded, with XX representing a character code in hexadecimal. This decoding takes place after the *path* has been split into components, but before each component is used in the equivalent of a **cd** or **get** command. Some often-used codes are ‘%2F’ (which represents ‘/’) and ‘%7E’ (which represents ‘~’).

The above interpretation has the following consequences:

- The path is interpreted relative to the default login directory of the specified user or of the ‘anonymous’ user. If the / directory is required, use a leading path of ‘%2F’. If a user’s home directory is required (and the remote server supports the syntax), use a leading path of ‘%7Euser/’. For example, to retrieve /etc/motd from ‘localhost’ as the user ‘myname’ with the password ‘mypass’, use ‘**ftp://myname:mypass@localhost/%2fetc/motd**’
- The exact **cd** and **get** commands can be controlled by careful choice of where to use ‘/’ and where to use ‘%2F’ (or ‘%2f’). For example, the following URLs correspond to the equivalents of the indicated commands:

<code>ftp://host/dir1/dir2/file</code>	“cd dir1”, “cd dir2”, “get file”.
<code>ftp://host/%2Fdir1/dir2/file</code>	“cd /dir1”, “cd dir2”, “get file”.
<code>ftp://host/dir1%2Fdir2/file</code>	“cd dir1/dir2”, “get file”.
<code>ftp://host/%2Fdir1%2Fdir2/file</code>	“cd /dir1/dir2”, “get file”.
<code>ftp://host/dir1%2Fdir2%2Ffile</code>	“get dir1/dir2/file”.
<code>ftp://host/%2Fdir1%2Fdir2%2Ffile</code>	“get /dir1/dir2/file”.

- You must have appropriate access permission for each of the intermediate directories that is used in the equivalent of a **cd** command.

http://[user[:password]@]host[:port]/path

An HTTP URL, retrieved using the HTTP protocol. If **set http_proxy** is defined, it is used as a URL to an HTTP proxy server. If HTTP authorization is required to retrieve *path*, and *user* (and optionally *password*) is in the URL, use them for the first attempt to authenticate.

https://[user[:password]@]host[:port]/path

An HTTPS URL, retrieved using the HTTPS protocol. If **set https_proxy** is defined, it is used as a URL to an HTTPS proxy server. If HTTPS authorization is required to retrieve *path*, and *user* (and optionally *password*) is in the URL, use them for the first attempt to authenticate. There is currently no certificate validation and verification.

file:///path

A local URL, copied from */path* on the local host.

about:topic

Display information regarding *topic*; no file is retrieved for this auto-fetched element. Supported values include:

about:ftp Information about **tnftp**.

about:version The version of **tnftp**. Useful to provide when reporting problems.

Unless noted otherwise above, and **-o output** is not given, the file is stored in the current directory as the basename(1) of *path*. Note that if a HTTP redirect is received, the fetch is retried using the new target URL supplied by the server, with a corresponding new *path*. Using an explicit **-o output** is recommended, to avoid writing to unexpected file names.

If a classic format or an FTP URL format has a trailing ‘/’ or an empty *path* component, then **tnftp** will connect to the site and **cd** to the directory given as the path, and leave the user in interactive mode ready for further input. This will not work if **set ftp_proxy** is being used.

Direct HTTP transfers use HTTP 1.1. Proxied FTP and HTTP transfers use HTTP 1.0.

If **-R** is given, all auto-fetches that don’t go via the FTP or HTTP proxies will be restarted. For FTP, this is implemented by using **reget** instead of **get**. For HTTP, this is implemented by using the ‘Range: bytes=’ HTTP/1.1 directive.

If WWW or proxy WWW authentication is required, you will be prompted to enter a username and password to authenticate with.

When specifying IPv6 numeric addresses in a URL, you need to surround the address in square brackets. E.g.: ‘**ftp://[::1]:21/**’. This is because colons are used in IPv6 numeric address as well as being the separator for the port number.

ABORTING A FILE TRANSFER

To abort a file transfer, use the terminal interrupt key (usually Ctrl-C). Sending transfers will be immediately halted. Receiving transfers will be halted by sending an FTP protocol ABOR command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server’s support for ABOR processing. If the remote server does not support the ABOR command, the prompt will not appear until the remote server has completed sending the requested file.

If the terminal interrupt key sequence is used whilst **tnftp** is awaiting a reply from the remote server for the ABOR processing, then the connection will be closed. This is different from the traditional behaviour (which ignores the terminal interrupt during this phase), but is considered more useful.

FILE NAMING CONVENTIONS

Files specified as arguments to **tnftp** commands are processed according to the following rules.

1. If the file name ‘-’ is specified, the *stdin* (for reading) or *stdout* (for writing) is used.
2. If the first character of the file name is ‘|’, the remainder of the argument is interpreted as a shell command. **tnftp** then forks a shell, using **popen(3)** with the argument supplied, and reads (writes) from the *stdout* (*stdin*). If the shell command includes spaces, the argument must be quoted; e.g.

- `"" | ls -lt"'. A particularly useful example of this mechanism is: 'dir "" | more'.`
3. Failing the above checks, if globbing is enabled, local file names are expanded according to the rules used in the `csh(1)`; see the `glob` command. If the `tnftp` command expects a single local file (e.g. `put`), only the first filename generated by the globbing operation is used.
 4. For `mget` commands and `get` commands with unspecified local file names, the local filename is the remote filename, which may be altered by a `case`, `ntrans`, or `nmap` setting. The resulting filename may then be altered if `runique` is on.
 5. For `mput` commands and `put` commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a `ntrans` or `nmap` setting. The resulting filename may then be altered by the remote server if `sunique` is on.

FILE TRANSFER PARAMETERS

The FTP specification specifies many parameters which may affect a file transfer. The `type` may be one of “ascii”, “image” (binary), “ebcdic”, and “local byte size” (for PDP-10’s and PDP-20’s mostly). `tnftp` supports the ascii and image types of file transfer, plus local byte size 8 for `tenex` mode transfers.

`tnftp` supports only the default values for the remaining file transfer parameters: `mode`, `form`, and `struct`.

THE .netrc FILE

The `.netrc` file contains login and initialization information used by the auto-login process. It resides in the user’s home directory, unless overridden with the `-N netrc` option, or specified in the NETRC environment variable. The following tokens are recognized; they may be separated by spaces, tabs, or new-lines:

`machine name`

Identify a remote machine `name`. The auto-login process searches the `.netrc` file for a `machine` token that matches the remote machine specified on the `tnftp` command line or as an `open` command argument. Once a match is made, the subsequent `.netrc` tokens are processed, stopping when the end of file is reached or another `machine` or a `default` token is encountered.

`default` This is the same as `machine name` except that `default` matches any name. There can be only one `default` token, and it must be after all `machine` tokens. This is normally used as:

```
default login anonymous password user@site
```

thereby giving the user an automatic anonymous FTP login to machines not specified in `.netrc`. This can be overridden by using the `-n` flag to disable auto-login.

`login name`

Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified `name`.

`password string`

Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note that if this token is present in the `.netrc` file for any user other than `anonymous`, `tnftp` will abort the auto-login process if the `.netrc` is readable by anyone besides the user.

`account string`

Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

macdef *name*

Define a macro. This token functions like the **tnftp macdef** command functions. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a blank line (consecutive new-line characters) is encountered. Like the other tokens in the .netrc file, a **macdef** is applicable only to the **machine** definition preceding it. A **macdef** entry cannot be used by multiple **machine** definitions; rather, it must be defined following each **machine** it is intended to be used with. If a macro named **init** is defined, it is automatically executed as the last step in the auto-login process. For example,

```
default
macdef init
epsv4 off
```

followed by a blank line.

COMMAND LINE EDITING

tnftp supports interactive command line editing, via the **editline(3)** library. It is enabled with the **edit** command, and is enabled by default if input is from a tty. Previous lines can be recalled and edited with the arrow keys, and other GNU Emacs-style editing keys may be used as well.

The **editline(3)** library is configured with a .editrc file — refer to **editrc(5)** for more information.

An extra key binding is available to **tnftp** to provide context sensitive command and filename completion (including remote file completion). To use this, bind a key to the **editline(3)** command **ftp-complete**. By default, this is bound to the TAB key.

COMMAND LINE PROMPT

By default, **tnftp** displays a command line prompt of ‘**ftp>**’ to the user. This can be changed with the **set prompt** command.

A prompt can be displayed on the right side of the screen (after the command input) with the **set rprompt** command.

The following formatting sequences are replaced by the given information:

%/ The current remote working directory.

%c[[0]n], %.[[0]n]

The trailing component of the current remote working directory, or *n* trailing components if a digit *n* is given. If *n* begins with ‘0’, the number of skipped components precede the trailing component(s) in the format “/⟨number⟩trailing” (for ‘%c’) or “...trailing” (for ‘%.’).

%M The remote host name.

%m The remote host name, up to the first dot ‘.’.

%n The remote user name.

%% A single percent character ‘%’.

ENVIRONMENT

tnftp uses the following environment variables.

FTPANONPASS Password to send in an anonymous FTP transfer. Defaults to ``whoami`@’.

FTPMODE Overrides the default operation mode. Support values are:

	active	active mode FTP only
	auto	automatic determination of passive or active (this is the default)
	gate	gate-ftp mode
	passive	passive mode FTP only
FTPPROMPT		Command-line prompt to use. Defaults to ‘ftp>’. Refer to COMMAND LINE PROMPT for more information.
FTPRPROMPT		Command-line right side prompt to use. Defaults to empty string. Refer to COMMAND LINE PROMPT for more information.
FTPSERVER		Host to use as gate-ftp server when gate is enabled.
FTPSERVERPORT		Port to use when connecting to gate-ftp server when gate is enabled. Default is port returned by a <code>getservbyname(3)</code> lookup of “ftpgate/tcp”.
FTPUSERAGENT		The value to send for the HTTP User-Agent header.
HOME		For default location of a <code>.netrc</code> file, if one exists.
NETRC		An alternate location of the <code>.netrc</code> file.
PAGER		Used by various commands to display files. Defaults to <code>more(1)</code> if empty or not set.
SHELL		For default shell.
ftp_proxy		URL of FTP proxy to use when making FTP URL requests (if not defined, use the standard FTP protocol).
		See <code>http_proxy</code> for further notes about proxy use.
http_proxy		URL of HTTP proxy to use when making HTTP URL requests. If proxy authentication is required and there is a username and password in this URL, they will automatically be used in the first attempt to authenticate to the proxy.
		If “unsafe” URL characters are required in the username or password (for example ‘@’ or ‘/’), encode them with RFC 3986 ‘%XX’ encoding.
		Note that the use of a username and password in <code>ftp_proxy</code> and <code>http_proxy</code> may be incompatible with other programs that use it (such as <code>lynx(1)</code>).
		<i>NOTE:</i> this is not used for interactive sessions, only for command-line fetches.
https_proxy		URL of HTTPS proxy to use when making HTTPS URL requests.
		See <code>http_proxy</code> for further notes about proxy use.
no_proxy		A space or comma separated list of hosts (or domains) for which proxying is not to be used. Each entry may have an optional trailing ‘:port’, which restricts the matching to connections to that port.

EXTENDED PASSIVE MODE AND FIREWALLS

Some firewall configurations do not allow `tnftp` to use extended passive mode. If you find that even a simple `ls` appears to hang after printing a message such as this:

```
229 Entering Extended Passive Mode ( ||| 58551 | )
```

then you will need to disable extended passive mode with `epsv4 off`. See the above section **The .netrc File** for an example of how to make this automatic.

SEE ALSO

`getservbyname(3)`, `editrc(5)`, `services(5)`, `ftpd(8)`

STANDARDS

`tnftp` attempts to be compliant with:

RFC 959

File Transfer Protocol

RFC 1123

Requirements for Internet Hosts - Application and Support

RFC 1635

How to Use Anonymous FTP

RFC 2389

Feature negotiation mechanism for the File Transfer Protocol

RFC 2428

FTP Extensions for IPv6 and NATs

RFC 2616

Hypertext Transfer Protocol -- HTTP/1.1

RFC 2822

Internet Message Format

RFC 3659

Extensions to FTP

RFC 3986

Uniform Resource Identifier (URI)

HISTORY

The `tnftp` command appeared in 4.2BSD.

Various features such as command line editing, context sensitive command and file completion, dynamic progress bar, automatic fetching of files and URLs, modification time preservation, transfer rate throttling, configurable command line prompt, and other enhancements over the standard BSD `tnftp` were implemented in NetBSD 1.3 and later releases by Luke Mewburn <lukem@NetBSD.org>.

IPv6 support was added by the WIDE/KAME project (but may not be present in all non-NetBSD versions of this program, depending if the operating system supports IPv6 in a similar manner to KAME).

BUGS

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2BSD ascii-mode transfer code has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2BSD servers using the ascii type. Avoid this problem by using the binary image type.

`tnftp` assumes that all IPv4 mapped addresses (IPv6 addresses with a form like `::ffff:10.1.1.1`) indicate IPv4 destinations which can be handled by `AF_INET` sockets. However, in certain IPv6 network configurations, this assumption is not true. In such an environment, IPv4 mapped addresses must be passed to `AF_INET6` sockets directly. For example, if your site uses a SIIT translator for IPv6-to-IPv4 translation, `tnftp` is unable to support your configuration.

NAME

getent – get entries from Name Service Switch libraries

SYNOPSIS

getent [option]... database key...

DESCRIPTION

The **getent** command displays entries from databases supported by the Name Service Switch libraries, which are configured in */etc/nsswitch.conf*. If one or more *key* arguments are provided, then only the entries that match the supplied keys will be displayed. Otherwise, if no *key* is provided, all entries will be displayed (unless the database does not support enumeration).

The *database* may be any of those supported by the GNU C Library, listed below:

ahosts	When no <i>key</i> is provided, use sethostent(3) , gethostent(3) , and endhostent(3) to enumerate the hosts database. This is identical to using hosts . When one or more <i>key</i> arguments are provided, pass each <i>key</i> in succession to getaddrinfo(3) with the address family AF_UNSPEC , enumerating each socket address structure returned.
ahostsv4	Same as ahosts , but use the address family AF_INET .
ahostsv6	Same as ahosts , but use the address family AF_INET6 . The call to getaddrinfo(3) in this case includes the AI_V4MAPPED flag.
aliases	When no <i>key</i> is provided, use setaliasent(3) , getaliasent(3) , and endaliasent(3) to enumerate the aliases database. When one or more <i>key</i> arguments are provided, pass each <i>key</i> in succession to getaliasbyname(3) and display the result.
ethers	When one or more <i>key</i> arguments are provided, pass each <i>key</i> in succession to ether_aton(3) and ether_ntohton(3) until a result is obtained, and display the result. Enumeration is not supported on ethers , so a <i>key</i> must be provided.
group	When no <i>key</i> is provided, use setgrent(3) , getgrent(3) , and endgrent(3) to enumerate the group database. When one or more <i>key</i> arguments are provided, pass each numeric <i>key</i> to getgrgid(3) and each nonnumeric <i>key</i> to getgrnam(3) and display the result.
gshadow	When no <i>key</i> is provided, use setsent(3) , getsent(3) , and endsent(3) to enumerate the gshadow database. When one or more <i>key</i> arguments are provided, pass each <i>key</i> in succession to getsgnam(3) and display the result.
hosts	When no <i>key</i> is provided, use sethostent(3) , gethostent(3) , and endhostent(3) to enumerate the hosts database. When one or more <i>key</i> arguments are provided, pass each <i>key</i> to gethostbyaddr(3) or gethostbyname2(3) , depending on whether a call to inet_pton(3) indicates that the <i>key</i> is an IPv6 or IPv4 address or not, and display the result.
initgroups	When one or more <i>key</i> arguments are provided, pass each <i>key</i> in succession to getgrouplist(3) and display the result. Enumeration is not supported on initgroups , so a <i>key</i> must be provided.
netgroup	When one <i>key</i> is provided, pass the <i>key</i> to setnetgrent(3) and, using getnetgrent(3) display the resulting string triple (<i>hostname</i> , <i>username</i> , <i>domainname</i>). Alternatively, three <i>keys</i> may be provided, which are interpreted as the <i>hostname</i> , <i>username</i> , and <i>domainname</i> to match to a netgroup name via innetgr(3) . Enumeration is not supported on netgroup , so either one or three <i>keys</i> must be provided.
networks	When no <i>key</i> is provided, use setnetent(3) , getnetent(3) , and endnetent(3) to enumerate the networks database. When one or more <i>key</i> arguments are provided, pass each numeric <i>key</i> to getnetbyaddr(3) and each nonnumeric <i>key</i> to getnetbyname(3) and display the result.
passwd	When no <i>key</i> is provided, use setpwent(3) , getpwent(3) , and endpwent(3) to enumerate the passwd database. When one or more <i>key</i> arguments are provided, pass each numeric <i>key</i> to getpwuid(3) and each nonnumeric <i>key</i> to getpwnam(3) and display the result.

- protocols** When no *key* is provided, use **setprotoent(3)**, **getprotoent(3)**, and **endprotoent(3)** to enumerate the protocols database. When one or more *key* arguments are provided, pass each numeric *key* to **getprotobynumber(3)** and each nonnumeric *key* to **getprotobynam(3)** and display the result.
- rpc** When no *key* is provided, use **setrpcent(3)**, **getrpcent(3)**, and **endrpcent(3)** to enumerate the rpc database. When one or more *key* arguments are provided, pass each numeric *key* to **getrpcbynumber(3)** and each nonnumeric *key* to **getrpcbynname(3)** and display the result.
- services** When no *key* is provided, use **setservent(3)**, **getservent(3)**, and **endservent(3)** to enumerate the services database. When one or more *key* arguments are provided, pass each numeric *key* to **getservbynumber(3)** and each nonnumeric *key* to **getservbynname(3)** and display the result.
- shadow** When no *key* is provided, use **setspent(3)**, **getspent(3)**, and **endspent(3)** to enumerate the shadow database. When one or more *key* arguments are provided, pass each *key* in succession to **getspnam(3)** and display the result.

OPTIONS

-s *service*, **--service** *service*

Override all databases with the specified service. (Since glibc 2.2.5.)

-s *database:service*, **--service** *database:service*

Override only specified databases with the specified service. The option may be used multiple times, but only the last service for each database will be used. (Since glibc 2.4.)

-i, **--no-idn**

Disables IDN encoding in lookups for **ahosts/getaddrinfo(3)** (Since glibc-2.13.)

-?, --help

Print a usage summary and exit.

--usage

Print a short usage summary and exit.

-V, **--version**

Print the version number, license, and disclaimer of warranty for **getent**.

EXIT STATUS

One of the following exit values can be returned by **getent**:

- 0** Command completed successfully.
- 1** Missing arguments, or *database* unknown.
- 2** One or more supplied *key* could not be found in the *database*.
- 3** Enumeration not supported on this *database*.

SEE ALSO

nsswitch.conf(5)

NAME

getfacl – get file access control lists

SYNOPSIS

getfacl [**-aceEsRLPtpndvh**] file ...

getfacl [**-aceEsRLPtpndvh**] –

DESCRIPTION

For each file, **getfacl** displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, **getfacl** also displays the default ACL. Non-directories cannot have default ACLs.

If **getfacl** is used on a file system that does not support ACLs, **getfacl** displays the access permissions defined by the traditional file mode permission bits.

The output format of **getfacl** is as follows:

```

1: # file: somedir/
2: # owner: lisa
3: # group: staff
4: # flags: -s-
5: user::rwx
6: user:joe:rwx          #effective:r-x
7: group::rwx            #effective:r-x
8: group:cool:r-x
9: mask::r-x
10: other::r-x
11: default:user::rwx
12: default:user:joe:rwx      #effective:r-x
13: default:group::r-x
14: default:mask::r-x
15: default:other::---
```

Lines 1–3 indicate the file name, owner, and owning group.

Line 4 indicates the setuid (s), setgid (s), and sticky (t) bits: either the letter representing the bit, or else a dash (–). This line is included if any of those bits is set and left out otherwise, so it will not be shown for most files. (See CONFORMANCE TO POSIX 1003.1e DRAFT STANDARD 17 below.)

Lines 5, 7 and 10 correspond to the user, group and other fields of the file mode permission bits. These three are called the base ACL entries. Lines 6 and 8 are named user and named group entries. Line 9 is the effective rights mask. This entry limits the effective rights granted to all groups and to named users. (The file owner and others permissions are not affected by the effective rights mask; all other entries are.) Lines 11–15 display the default ACL associated with this directory. Directories may have a default ACL. Regular files never have a default ACL.

The default behavior for **getfacl** is to display both the ACL and the default ACL, and to include an effective rights comment for lines where the rights of the entry differ from the effective rights.

If output is to a terminal, the effective rights comment is aligned to column 40. Otherwise, a single tab character separates the ACL entry and the effective rights comment.

The ACL listings of multiple files are separated by blank lines. The output of **getfacl** can also be used as input to **setfacl**.

PERMISSIONS

Process with search access to a file (i.e., processes with read access to the containing directory of a file) are also granted read access to the file's ACLs. This is analogous to the permissions required for accessing the file mode.

OPTIONS

- a, --access*
Display the file access control list.
- d, --default*
Display the default access control list.
- c, --omit-header*
Do not display the comment header (the first three lines of each file's output).
- e, --all-effective*
Print all effective rights comments, even if identical to the rights defined by the ACL entry.
- E, --no-effective*
Do not print effective rights comments.
- s, --skip-base*
Skip files that only have the base ACL entries (owner, group, others).
- R, --recursive*
List the ACLs of all files and directories recursively.
- L, --logical*
Logical walk, follow symbolic links to directories. The default behavior is to follow symbolic link arguments, and skip symbolic links encountered in subdirectories. Only effective in combination with *-R*.
- P, --physical*
Physical walk, do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with *-R*.
- t, --tabular*
Use an alternative tabular output format. The ACL and the default ACL are displayed side by side. Permissions that are ineffective due to the ACL mask entry are displayed capitalized. The entry tag names for the ACL_USER_OBJ and ACL_GROUP_OBJ entries are also displayed in capital letters, which helps in spotting those entries.
- p, --absolute-names*
Do not strip leading slash characters ('/'). The default behavior is to strip leading slash characters.
- n, --numeric*
List numeric user and group IDs
- v, --version*
Print the version of getfacl and exit.
- h, --help*
Print help explaining the command line options.
- End of command line options. All remaining parameters are interpreted as file names, even if they start with a dash character.
- If the file name parameter is a single dash character, getfacl reads a list of files from standard input.

CONFORMANCE TO POSIX 1003.1e DRAFT STANDARD 17

If the environment variable `POSIXLY_CORRECT` is defined, the default behavior of `getfacl` changes in the following ways: Unless otherwise specified, only the ACL is printed. The default ACL is only printed if the *-d* option is given. If no command line parameter is given, `getfacl` behaves as if it was invoked as "getfacl

–”. No flags comments indicating the setuid, setgid, and sticky bits are generated.

AUTHOR

Andreas Gruenbacher, <*andreas.gruenbacher@gmail.com*>.

Please send your bug reports and comments to the above address.

SEE ALSO

setfacl(1), acl(5)

NAME

getfattr – get extended attributes of filesystem objects

SYNOPSIS

```
getfattr [-hRLP] -n name [-e en] pathname...
getfattr [-hRLP] -d [-e en] [-m pattern] pathname...
```

DESCRIPTION

For each file, **getfattr** displays the file name, and the set of extended attribute names (and optionally values) which are associated with that file. Per default only attributes in the user namespace are displayed, see **-m**.

The output format of **getfattr -d** is as follows:

```
1: # file: somedir/
2: user.name0="value0"
3: user.name1="value1"
4: user.name2="value2"
5: ...
```

Line 1 identifies the file name for which the following lines are being reported. The remaining lines (lines 2 to 4 above) show the *name* and *value* pairs associated with the specified file.

OPTIONS

-n name, --name=name

Dump the value of the named extended attribute.

-d, --dump

Dump the values of all matched extended attributes.

-e en, --encoding=en

Encode values after retrieving them. Valid values of *en* are "text", "hex", and "base64". Values encoded as text strings are enclosed in double quotes ("), while strings encoded as hexadecimal and base64 are prefixed with 0x and 0s, respectively.

-h, --no-dereference

Do not dereference symlinks. Instead of the file a symlink refers to, the symlink itself is examined. Unless doing a logical (-L) traversal, do not traverse symlinks to directories.

-m pattern, --match=pattern

Only include attributes with names matching the regular expression *pattern*. The default value for *pattern* is "^user\\.", which includes all the attributes in the user namespace. Specify "-" for including all attributes. Refer to **attr(5)** for a more detailed discussion of namespaces.

--absolute-names

Do not strip leading slash characters ('/'). The default behaviour is to strip leading slash characters.

--only-values

Dump out the raw extended attribute value(s) without encoding them.

-R, --recursive

List the attributes of all files and directories recursively.

-L, --logical

Logical walk, follow symbolic links to directories. The default behaviour is to follow symbolic link arguments unless **--no-dereference** is given, and to skip symbolic links encountered in subdirectories. Only effective in combination with **-R**.

-P, --physical

Physical walk, do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with **-R**.

--version

Print the version of **getfattr** and exit.

--help

Print help explaining the command line options.

-- End of command line options. All remaining parameters are interpreted as file names, even if they start with a dash character.

AUTHOR

Andreas Gruenbacher, <andreas.gruenbacher@gmail.com> and the SGI XFS development team, <linux-xfs@oss.sgi.com>.

Please send your bug reports or comments to <<https://savannah.nongnu.org/bugs/?group=attr>> or <acl-devel@nongnu.org>.

SEE ALSO

setfattr(1), attr(5)

NAME

getgroups, setgroups – get/set list of supplementary group IDs

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int getgroups(int size, gid_t list[]);
#include <grp.h>
int setgroups(size_t size, const gid_t *_Nullable list);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

```
setgroups():
  Since glibc 2.19:
    _DEFAULT_SOURCE
  glibc 2.19 and earlier:
    _BSD_SOURCE
```

DESCRIPTION

getgroups() returns the supplementary group IDs of the calling process in *list*. The argument *size* should be set to the maximum number of items that can be stored in the buffer pointed to by *list*. If the calling process is a member of more than *size* supplementary groups, then an error results.

It is unspecified whether the effective group ID of the calling process is included in the returned list. (Thus, an application should also call **getegid**(2) and add or remove the resulting value.)

If *size* is zero, *list* is not modified, but the total number of supplementary group IDs for the process is returned. This allows the caller to determine the size of a dynamically allocated *list* to be used in a further call to **getgroups()**.

setgroups() sets the supplementary group IDs for the calling process. Appropriate privileges are required (see the description of the **EPERM** error, below). The *size* argument specifies the number of supplementary group IDs in the buffer pointed to by *list*. A process can drop all of its supplementary groups with the call:

```
setgroups(0, NULL);
```

RETURN VALUE

On success, **getgroups()** returns the number of supplementary group IDs. On error, -1 is returned, and *errno* is set to indicate the error.

On success, **setgroups()** returns 0. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EFAULT

list has an invalid address.

getgroups() can additionally fail with the following error:

EINVAL

size is less than the number of supplementary group IDs, but is not zero.

setgroups() can additionally fail with the following errors:

EINVAL

size is greater than **NGROUPS_MAX** (32 before Linux 2.6.4; 65536 since Linux 2.6.4).

ENOMEM

Out of memory.

EPERM

The calling process has insufficient privilege (the caller does not have the **CAP_SETGID** capability in the user namespace in which it resides).

EPERM (since Linux 3.19)

The use of **setgroups()** is denied in this user namespace. See the description of */proc/pid/setgroups* in **user_namespaces(7)**.

STANDARDS

getgroups(): SVr4, 4.3BSD, POSIX.1-2001, POSIX.1-2008.

setgroups(): SVr4, 4.3BSD. Since **setgroups()** requires privilege, it is not covered by POSIX.1.

NOTES

A process can have up to **NGROUPS_MAX** supplementary group IDs in addition to the effective group ID. The constant **NGR_OUPS_MAX** is defined in *<limits.h>*. The set of supplementary group IDs is inherited from the parent process, and preserved across an **execve(2)**.

The maximum number of supplementary group IDs can be found at run time using **sysconf(3)**:

```
long ngroups_max;
ngroups_max = sysconf(_SC_NGROUPS_MAX);
```

The maximum return value of **getgroups()** cannot be larger than one more than this value. Since Linux 2.6.4, the maximum number of supplementary group IDs is also exposed via the Linux-specific read-only file, */proc/sys/kernel/ngroups_max*.

The original Linux **getgroups()** system call supported only 16-bit group IDs. Subsequently, Linux 2.4 added **getgroups32()**, supporting 32-bit IDs. The glibc **getgroups()** wrapper function transparently deals with the variation across kernel versions.

C library/kernel differences

At the kernel level, user IDs and group IDs are a per-thread attribute. However, POSIX requires that all threads in a process share the same credentials. The NPTL threading implementation handles the POSIX requirements by providing wrapper functions for the various system calls that change process UIDs and GIDs. These wrapper functions (including the one for **setgroups()**) employ a signal-based technique to ensure that when one thread changes credentials, all of the other threads in the process also change their credentials. For details, see **nptl(7)**.

SEE ALSO

getgid(2), **setgid(2)**, **getgrouplist(3)**, **group_member(3)**, **initgroups(3)**, **capabilities(7)**, **credentials(7)**

NAME

getgroups, setgroups – get/set list of supplementary group IDs

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int getgroups(int size, gid_t list[]);
#include <grp.h>
int setgroups(size_t size, const gid_t *_Nullable list);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

```
setgroups():
  Since glibc 2.19:
    _DEFAULT_SOURCE
  glibc 2.19 and earlier:
    _BSD_SOURCE
```

DESCRIPTION

getgroups() returns the supplementary group IDs of the calling process in *list*. The argument *size* should be set to the maximum number of items that can be stored in the buffer pointed to by *list*. If the calling process is a member of more than *size* supplementary groups, then an error results.

It is unspecified whether the effective group ID of the calling process is included in the returned list. (Thus, an application should also call **getegid**(2) and add or remove the resulting value.)

If *size* is zero, *list* is not modified, but the total number of supplementary group IDs for the process is returned. This allows the caller to determine the size of a dynamically allocated *list* to be used in a further call to **getgroups()**.

setgroups() sets the supplementary group IDs for the calling process. Appropriate privileges are required (see the description of the **EPERM** error, below). The *size* argument specifies the number of supplementary group IDs in the buffer pointed to by *list*. A process can drop all of its supplementary groups with the call:

```
setgroups(0, NULL);
```

RETURN VALUE

On success, **getgroups()** returns the number of supplementary group IDs. On error, -1 is returned, and *errno* is set to indicate the error.

On success, **setgroups()** returns 0. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EFAULT

list has an invalid address.

getgroups() can additionally fail with the following error:

EINVAL

size is less than the number of supplementary group IDs, but is not zero.

setgroups() can additionally fail with the following errors:

EINVAL

size is greater than **NGROUPS_MAX** (32 before Linux 2.6.4; 65536 since Linux 2.6.4).

ENOMEM

Out of memory.

EPERM

The calling process has insufficient privilege (the caller does not have the **CAP_SETGID** capability in the user namespace in which it resides).

EPERM (since Linux 3.19)

The use of **setgroups()** is denied in this user namespace. See the description of */proc/pid/setgroups* in **user_namespaces(7)**.

STANDARDS

getgroups(): SVr4, 4.3BSD, POSIX.1-2001, POSIX.1-2008.

setgroups(): SVr4, 4.3BSD. Since **setgroups()** requires privilege, it is not covered by POSIX.1.

NOTES

A process can have up to **NGROUPS_MAX** supplementary group IDs in addition to the effective group ID. The constant **NGR_OUPS_MAX** is defined in *<limits.h>*. The set of supplementary group IDs is inherited from the parent process, and preserved across an **execve(2)**.

The maximum number of supplementary group IDs can be found at run time using **sysconf(3)**:

```
long ngroups_max;
ngroups_max = sysconf(_SC_NGROUPS_MAX);
```

The maximum return value of **getgroups()** cannot be larger than one more than this value. Since Linux 2.6.4, the maximum number of supplementary group IDs is also exposed via the Linux-specific read-only file, */proc/sys/kernel/ngroups_max*.

The original Linux **getgroups()** system call supported only 16-bit group IDs. Subsequently, Linux 2.4 added **getgroups32()**, supporting 32-bit IDs. The glibc **getgroups()** wrapper function transparently deals with the variation across kernel versions.

C library/kernel differences

At the kernel level, user IDs and group IDs are a per-thread attribute. However, POSIX requires that all threads in a process share the same credentials. The NPTL threading implementation handles the POSIX requirements by providing wrapper functions for the various system calls that change process UIDs and GIDs. These wrapper functions (including the one for **setgroups()**) employ a signal-based technique to ensure that when one thread changes credentials, all of the other threads in the process also change their credentials. For details, see **nptl(7)**.

SEE ALSO

getgid(2), **setgid(2)**, **getgrouplist(3)**, **group_member(3)**, **initgroups(3)**, **capabilities(7)**, **credentials(7)**

NAME

`gethostname`, `sethostname` – get/set hostname

LIBRARY

Standard C library (`libc`, `-lc`)

SYNOPSIS

```
#include <unistd.h>
int gethostname(char *name, size_t len);
int sethostname(const char *name, size_t len);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

```
gethostname():
_XOPEN_SOURCE >= 500 || _POSIX_C_SOURCE >= 200112L
|| /* glibc 2.19 and earlier */ _BSD_SOURCE

sethostname():
Since glibc 2.21:
_DEFAULT_SOURCE
In glibc 2.19 and 2.20:
_DEFAULT_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
Up to and including glibc 2.19:
_BSD_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
```

DESCRIPTION

These system calls are used to access or to change the system hostname. More precisely, they operate on the hostname associated with the calling process's UTS namespace.

sethostname() sets the hostname to the value given in the character array *name*. The *len* argument specifies the number of bytes in *name*. (Thus, *name* does not require a terminating null byte.)

gethostname() returns the null-terminated hostname in the character array *name*, which has a length of *len* bytes. If the null-terminated hostname is too large to fit, then the name is truncated, and no error is returned (but see NOTES below). POSIX.1 says that if such truncation occurs, then it is unspecified whether the returned buffer includes a terminating null byte.

RETURN VALUE

On success, zero is returned. On error, `-1` is returned, and *errno* is set to indicate the error.

ERRORS**FAULT**

name is an invalid address.

EINVAL

len is negative or, for **sethostname()**, *len* is larger than the maximum allowed size.

ENAMETOOLONG

(glibc **gethostname()**) *len* is smaller than the actual size. (Before glibc 2.1, glibc uses **EINVAL** for this case.)

EPERM

For **sethostname()**, the caller did not have the **CAP_SYS_ADMIN** capability in the user namespace associated with its UTS namespace (see **namespaces**(7)).

STANDARDS

SVr4, 4.4BSD (these interfaces first appeared in 4.2BSD). POSIX.1-2001 and POSIX.1-2008 specify **gethostname()** but not **sethostname()**.

NOTES

SUSv2 guarantees that "Host names are limited to 255 bytes". POSIX.1 guarantees that "Host names (not including the terminating null byte) are limited to **HOST_NAME_MAX** bytes". On Linux, **HOST_NAME_MAX** is defined with the value 64, which has been the limit since Linux 1.0 (earlier

kernels imposed a limit of 8 bytes).

C library/kernel differences

The GNU C library does not employ the **gethostname()** system call; instead, it implements **gethostname()** as a library function that calls **uname(2)** and copies up to *len* bytes from the returned *nodename* field into *name*. Having performed the copy, the function then checks if the length of the *nodename* was greater than or equal to *len*, and if it is, then the function returns -1 with *errno* set to **ENAMETOOLONG**; in this case, a terminating null byte is not included in the returned *name*.

Versions of glibc before glibc 2.2 handle the case where the length of the *nodename* was greater than or equal to *len* differently: nothing is copied into *name* and the function returns -1 with *errno* set to **ENAME-TOOLONG**.

SEE ALSO

hostname(1), **getdomainname(2)**, **setdomainname(2)**, **uname(2)**, **uts_namespaces(7)**

NAME

getxattr, lgetxattr, fgetxattr – retrieve an extended attribute value

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/xattr.h>

ssize_t getxattr(const char *path, const char *name,
                 void value[.size], size_t size);
ssize_t lgetxattr(const char *path, const char *name,
                  void value[.size], size_t size);
ssize_t fgetxattr(int fd, const char *name,
                  void value[.size], size_t size);
```

DESCRIPTION

Extended attributes are *name:value* pairs associated with inodes (files, directories, symbolic links, etc.). They are extensions to the normal attributes which are associated with all inodes in the system (i.e., the **stat(2)** data). A complete overview of extended attributes concepts can be found in **xattr(7)**.

getxattr() retrieves the value of the extended attribute identified by *name* and associated with the given *path* in the filesystem. The attribute value is placed in the buffer pointed to by *value*; *size* specifies the size of that buffer. The return value of the call is the number of bytes placed in *value*.

lgetxattr() is identical to **getxattr()**, except in the case of a symbolic link, where the link itself is interrogated, not the file that it refers to.

fgetxattr() is identical to **getxattr()**, only the open file referred to by *fd* (as returned by **open(2)**) is interrogated in place of *path*.

An extended attribute *name* is a null-terminated string. The name includes a namespace prefix; there may be several, disjoint namespaces associated with an individual inode. The value of an extended attribute is a chunk of arbitrary textual or binary data that was assigned using **setxattr(2)**.

If *size* is specified as zero, these calls return the current size of the named extended attribute (and leave *value* unchanged). This can be used to determine the size of the buffer that should be supplied in a subsequent call. (But, bear in mind that there is a possibility that the attribute value may change between the two calls, so that it is still necessary to check the return status from the second call.)

RETURN VALUE

On success, these calls return a nonnegative value which is the size (in bytes) of the extended attribute value. On failure, -1 is returned and *errno* is set to indicate the error.

ERRORS

E2BIG The size of the attribute value is larger than the maximum size allowed; the attribute cannot be retrieved. This can happen on filesystems that support very large attribute values such as NFSv4, for example.

ENODATA

The named attribute does not exist, or the process has no access to this attribute.

ENOTSUP

Extended attributes are not supported by the filesystem, or are disabled.

ERANGE

The *size* of the *value* buffer is too small to hold the result.

In addition, the errors documented in **stat(2)** can also occur.

VERSIONS

These system calls have been available since Linux 2.4; glibc support is provided since glibc 2.3.

STANDARDS

These system calls are Linux-specific.

EXAMPLES

See **listxattr(2)**.

SEE ALSO

getattr(1), **setattr(1)**, **listxattr(2)**, **open(2)**, **removexattr(2)**, **setxattr(2)**, **stat(2)**, **symlink(7)**, **xattr(7)**

NAME

gpasswd – administer /etc/group and /etc/gshadow

SYNOPSIS

gpasswd [*option*] *group*

DESCRIPTION

The **gpasswd** command is used to administer /etc/group, and /etc/gshadow. Every group can have administrators, members and a password.

System administrators can use the **-A** option to define group administrator(s) and the **-M** option to define members. They have all rights of group administrators and members.

gpasswd called by a group administrator with a group name only prompts for the new password of the *group*.

If a password is set the members can still use **newgrp(1)** without a password, and non-members must supply the password.

Notes about group passwords

Group passwords are an inherent security problem since more than one person is permitted to know the password. However, groups are a useful tool for permitting co-operation between different users.

OPTIONS

Except for the **-A** and **-M** options, the options cannot be combined.

The options which apply to the **gpasswd** command are:

-a, --add *user*

Add the *user* to the named *group*.

-d, --delete *user*

Remove the *user* from the named *group*.

-h, --help

Display help message and exit.

-Q, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-r, --remove-password

Remove the password from the named *group*. The group password will be empty. Only group members will be allowed to use **newgrp** to join the named *group*.

-R, --restrict

Restrict the access to the named *group*. The group password is set to "!". Only group members with a password will be allowed to use **newgrp** to join the named *group*.

-A, --administrators *user*,...

Set the list of administrative users.

-M, --members *user*,...

Set the list of group members.

CAVEATS

This tool only operates on the /etc/group and /etc/gshadow files. Thus you cannot change any NIS or LDAP group. This must be performed on the corresponding server.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

ENCRYPT_METHOD (string)

This defines the system default encryption algorithm for encrypting passwords (if no algorithm are specified on the command line).

It can take one of these values: *DES* (default), *MD5*, *SHA256*, *SHA512*.

Note: this parameter overrides the **MD5_CRYPT_ENAB** variable.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

MD5_CRYPT_ENAB (boolean)

Indicate if passwords must be encrypted using the MD5-based algorithm. If set to *yes*, new passwords will be encrypted using the MD5-based algorithm compatible with the one used by recent releases of FreeBSD. It supports passwords of unlimited length and longer salt strings. Set to *no* if you need to copy encrypted passwords to other systems which don't understand the new algorithm. Default is *no*.

This variable is superseded by the **ENCRYPT_METHOD** variable or by any command line option used to configure the encryption algorithm.

This variable is deprecated. You should use **ENCRYPT_METHOD**.

Note: This only affect the generation of group passwords. The generation of user passwords is done by PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

SHA_CRYPT_MIN_ROUNDS (number), **SHA_CRYPT_MAX_ROUNDS** (number)

When **ENCRYPT_METHOD** is set to *SHA256* or *SHA512*, this defines the number of SHA rounds used by the encryption algorithm by default (when the number of rounds is not specified on the command line).

With a lot of rounds, it is more difficult to brute forcing the password. But note also that more CPU resources will be needed to authenticate users.

If not specified, the libc will choose the default number of rounds (5000).

The values must be inside the 1000–999,999,999 range.

If only one of the **SHA_CRYPT_MIN_ROUNDS** or **SHA_CRYPT_MAX_ROUNDS** values is set, then this value will be used.

If **SHA_CRYPT_MIN_ROUNDS > SHA_CRYPT_MAX_ROUNDS**, the highest value will be used.

Note: This only affect the generation of group passwords. The generation of user passwords is done by

PAM and subject to the PAM configuration. It is recommended to set this variable consistently with the PAM configuration.

FILES

- /etc/group
Group account information.
- /etc/gshadow
Secure group account information.

SEE ALSO

- newgrp(1), groupadd(8), groupdel(8), groupmod(8), grpck(8), group(5), gshadow(5).**

NAME

grep, **egrep**, **fgrep**, **rgrep** – print lines that match patterns

SYNOPSIS

```
grep [OPTION...] PATTERNs [FILE...]
grep [OPTION...] -e PATTERNs ... [FILE...]
grep [OPTION...] -f PATTERN_FILE ... [FILE...]
```

DESCRIPTION

grep searches for *PATTERNs* in each *FILE*. *PATTERNs* is one or more patterns separated by newline characters, and **grep** prints each line that matches a pattern. Typically *PATTERNs* should be quoted when **grep** is used in a shell command.

A *FILE* of “–” stands for standard input. If no *FILE* is given, recursive searches examine the working directory, and nonrecursive searches read standard input.

In addition, the variant programs **egrep**, **fgrep** and **rgrep** are the same as **grep -E**, **grep -F**, and **grep -r**, respectively. These variants are deprecated, but are provided for backward compatibility.

OPTIONS**Generic Program Information**

--help Output a usage message and exit.

-V, --version

Output the version number of **grep** and exit.

Pattern Syntax

-E, --extended-regexp

Interpret *PATTERNs* as extended regular expressions (EREs, see below).

-F, --fixed-strings

Interpret *PATTERNs* as fixed strings, not regular expressions.

-G, --basic-regexp

Interpret *PATTERNs* as basic regular expressions (BREs, see below). This is the default.

-P, --perl-regexp

Interpret I<PATTERNs> as Perl-compatible regular expressions (PCREs). This option is experimental when combined with the **-z** (**--null-data**) option, and **grep -P** may warn of unimplemented features.

Matching Control

-e PATTERNs, --regexp=PATTERNs

Use *PATTERNs* as the patterns. If this option is used multiple times or is combined with the **-f** (**--file**) option, search for all patterns given. This option can be used to protect a pattern beginning with “–”.

-f FILE, --file=FILE

Obtain patterns from *FILE*, one per line. If this option is used multiple times or is combined with the **-e** (**--regexp**) option, search for all patterns given. The empty file contains zero patterns, and therefore matches nothing.

-i, --ignore-case

Ignore case distinctions in patterns and input data, so that characters that differ only in case match each other.

--no-ignore-case

Do not ignore case distinctions in patterns and input data. This is the default. This option is useful for passing to shell scripts that already use **-i**, to cancel its effects because the two options override each other.

-v, --invert-match

Invert the sense of matching, to select non-matching lines.

-w, --word-regexp

Select only those lines containing matches that form whole words. The test is that the matching substring must either be at the beginning of the line, or preceded by a non-word constituent character. Similarly, it must be either at the end of the line or followed by a non-word constituent character. Word-constituent characters are letters, digits, and the underscore. This option has no effect if **-x** is also specified.

-x, --line-regexp

Select only those matches that exactly match the whole line. For a regular expression pattern, this is like parenthesizing the pattern and then surrounding it with ^ and \$.

-y Obsolete synonym for **-i**.**General Output Control****-c, --count**

Suppress normal output; instead print a count of matching lines for each input file. With the **-v**, **--invert-match** option (see below), count non-matching lines.

--color[=WHEN], --colour[=WHEN]

Surround the matched (non-empty) strings, matching lines, context lines, file names, line numbers, byte offsets, and separators (for fields and groups of context lines) with escape sequences to display them in color on the terminal. The colors are defined by the environment variable **GREP_COLORS**. The deprecated environment variable **GREP_COLOR** is still supported, but its setting does not have priority. *WHEN* is **never**, **always**, or **auto**.

-L, --files-without-match

Suppress normal output; instead print the name of each input file from which no output would normally have been printed.

-l, --files-with-matches

Suppress normal output; instead print the name of each input file from which output would normally have been printed. Scanning each input file stops upon first match.

-m NUM, --max-count=NUM

Stop reading a file after *NUM* matching lines. If the input is standard input from a regular file, and *NUM* matching lines are output, **grep** ensures that the standard input is positioned to just after the last matching line before exiting, regardless of the presence of trailing context lines. This enables a calling process to resume a search. When **grep** stops after *NUM* matching lines, it outputs any trailing context lines. When the **-c** or **--count** option is also used, **grep** does not output a count greater than *NUM*. When the **-v** or **--invert-match** option is also used, **grep** stops after outputting *NUM* non-matching lines.

-o, --only-matching

Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.

-q, --quiet, --silent

Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the **-s** or **--no-messages** option.

-s, --no-messages

Suppress error messages about nonexistent or unreadable files.

Output Line Prefix Control**-b, --byte-offset**

Print the 0-based byte offset within the input file before each line of output. If **-o** (**--only-matching**) is specified, print the offset of the matching part itself.

-H, --with-filename

Print the file name for each match. This is the default when there is more than one file to search. This is a GNU extension.

-h, --no-filename

Suppress the prefixing of file names on output. This is the default when there is only one file (or only standard input) to search.

--label=LABEL

Display input actually coming from standard input as input coming from file *LABEL*. This can be useful for commands that transform a file's contents before searching, e.g., `gzip -cd foo.gz | grep --label=foo -H 'some pattern'`. See also the **-H** option.

-n, --line-number

Prefix each line of output with the 1-based line number within its input file.

-T, --initial-tab

Make sure that the first character of actual line content lies on a tab stop, so that the alignment of tabs looks normal. This is useful with options that prefix their output to the actual content: **-H**, **-n**, and **-b**. In order to improve the probability that lines from a single file will all start at the same column, this also causes the line number and byte offset (if present) to be printed in a minimum size field width.

-Z, --null

Output a zero byte (the ASCII **NUL** character) instead of the character that normally follows a file name. For example, `grep -lZ` outputs a zero byte after each file name instead of the usual newline. This option makes the output unambiguous, even in the presence of file names containing unusual characters like newlines. This option can be used with commands like **find -print0**, **perl -0**, **sort -z**, and **xargs -0** to process arbitrary file names, even those that contain newline characters.

Context Line Control**-A NUM, --after-context=NUM**

Print *NUM* lines of trailing context after matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the **-o** or **--only-matching** option, this has no effect and a warning is given.

-B NUM, --before-context=NUM

Print *NUM* lines of leading context before matching lines. Places a line containing a group separator (--) between contiguous groups of matches. With the **-o** or **--only-matching** option, this has no effect and a warning is given.

-C NUM, -NUM, --context=NUM

Print *NUM* lines of output context. Places a line containing a group separator (--) between contiguous groups of matches. With the **-o** or **--only-matching** option, this has no effect and a warning is given.

--group-separator=SEP

When **-A**, **-B**, or **-C** are in use, print *SEP* instead of **--** between groups of lines.

--no-group-separator

When **-A**, **-B**, or **-C** are in use, do not print a separator between groups of lines.

File and Directory Selection**-a, --text**

Process a binary file as if it were text; this is equivalent to the **--binary-files=text** option.

--binary-files=TYPE

If a file's data or metadata indicate that the file contains binary data, assume that the file is of type *TYPE*. Non-text bytes indicate binary data; these are either output bytes that are improperly encoded for the current locale, or null input bytes when the **-z** option is not given.

By default, *TYPE* is **binary**, and **grep** suppresses output after null input binary data is discovered, and suppresses output lines that contain improperly encoded data. When some output is suppressed, **grep** follows any output with a one-line message saying that a binary file matches.

If *TYPE* is **without-match**, when **grep** discovers null input binary data it assumes that the rest of the file does not match; this is equivalent to the **-I** option.

If *TYPE* is **text**, **grep** processes a binary file as if it were text; this is equivalent to the **-a** option.

When *type* is **binary**, **grep** may treat non-text bytes as line terminators even without the **-z** option. This means choosing **binary** versus **text** can affect whether a pattern matches a file. For example, when *type* is **binary** the pattern **q\$** might match **q** immediately followed by a null byte, even though this is not matched when *type* is **text**. Conversely, when *type* is **binary** the pattern **.** (period) might not match a null byte.

Warning: The **-a** option might output binary garbage, which can have nasty side effects if the output is a terminal and if the terminal driver interprets some of it as commands. On the other hand, when reading files whose text encodings are unknown, it can be helpful to use **-a** or to set **LC_ALL='C'** in the environment, in order to find more matches even if the matches are unsafe for direct display.

-D ACTION, --devices=ACTION

If an input file is a device, FIFO or socket, use *ACTION* to process it. By default, *ACTION* is **read**, which means that devices are read just as if they were ordinary files. If *ACTION* is **skip**, devices are silently skipped.

-d ACTION, --directories=ACTION

If an input file is a directory, use *ACTION* to process it. By default, *ACTION* is **read**, i.e., read directories just as if they were ordinary files. If *ACTION* is **skip**, silently skip directories. If *ACTION* is **recurse**, read all files under each directory, recursively, following symbolic links only if they are on the command line. This is equivalent to the **-r** option.

--exclude=GLOB

Skip any command-line file with a name suffix that matches the pattern *GLOB*, using wildcard matching; a name suffix is either the whole name, or a trailing part that starts with a non-slash character immediately after a slash (/) in the name. When searching recursively, skip any subfile whose base name matches *GLOB*; the base name is the part after the last slash. A pattern can use *****, **?**, and **[...]** as wildcards, and **** to quote a wildcard or backslash character literally.

--exclude-from=FILE

Skip files whose base name matches any of the file-name globs read from *FILE* (using wildcard matching as described under **--exclude**).

--exclude-dir=GLOB

Skip any command-line directory with a name suffix that matches the pattern *GLOB*. When searching recursively, skip any subdirectory whose base name matches *GLOB*. Ignore any redundant trailing slashes in *GLOB*.

-I Process a binary file as if it did not contain matching data; this is equivalent to the **--binary-files=without-match** option.

--include=GLOB

Search only files whose base name matches *GLOB* (using wildcard matching as described under **--exclude**). If contradictory **--include** and **--exclude** options are given, the last matching one wins. If no **--include** or **--exclude** options match, a file is included unless the first such option is **--include**.

-r, --recursive

Read all files under each directory, recursively, following symbolic links only if they are on the command line. Note that if no file operand is given, B`<grep>` searches the working directory. This is equivalent to the **-d recurse** option.

-R, --dereference-recursive

Read all files under each directory, recursively. Follow all symbolic links, unlike **-r**.

Other Options**--line-buffered**

Use line buffering on output. This can cause a performance penalty.

-U, --binary

Treat the file(s) as binary. By default, under MS-DOS and MS-Windows, **grep** guesses whether a file is text or binary as described for the **--binary-files** option. If **grep** decides the file is a text file, it strips the CR characters from the original file contents (to make regular expressions with ^ and \$ work correctly). Specifying **-U** overrules this guesswork, causing all files to be read and passed to the matching mechanism verbatim; if the file is a text file with CR/LF pairs at the end of each line, this will cause some regular expressions to fail. This option has no effect on platforms other than MS-DOS and MS-Windows.

-z, --null-data

Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline. Like the **-Z** or **--null** option, this option can be used with commands like **sort -z** to process arbitrary file names.

REGULAR EXPRESSIONS

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.

grep understands three different versions of regular expression syntax: “basic” (BRE), “extended” (ERE) and “perl” (PCRE). In GNU **grep** there is no difference in available functionality between basic and extended syntaxes. In other implementations, basic regular expressions are less powerful. The following description applies to extended regular expressions; differences for basic regular expressions are summarized afterwards. Perl-compatible regular expressions give additional functionality, and are documented in B<pcresyntax>(3) and B<pcrepattern>(3), but work only if PCRE support is enabled.

The fundamental building blocks are the regular expressions that match a single character. Most characters, including all letters and digits, are regular expressions that match themselves. Any meta-character with special meaning may be quoted by preceding it with a backslash.

The period **.** matches any single character. It is unspecified whether it matches an encoding error.

Character Classes and Bracket Expressions

A *bracket expression* is a list of characters enclosed by [and]. It matches any single character in that list. If the first character of the list is the caret ^ then it matches any character *not* in the list; it is unspecified whether it matches an encoding error. For example, the regular expression **[0123456789]** matches any single digit.

Within a bracket expression, a *range expression* consists of two characters separated by a hyphen. It matches any single character that sorts between the two characters, inclusive, using the locale’s collating sequence and character set. For example, in the default C locale, **[a-d]** is equivalent to **[abcd]**. Many locales sort characters in dictionary order, and in these locales **[a-d]** is typically not equivalent to **[abcd]**; it might be equivalent to **[aBbCcDd]**, for example. To obtain the traditional interpretation of bracket expressions, you can use the C locale by setting the **LC_ALL** environment variable to the value **C**.

Finally, certain named classes of characters are predefined within bracket expressions, as follows. Their names are self explanatory, and they are **[:alnum:]**, **[:alpha:]**, **[:blank:]**, **[:cntrl:]**, **[:digit:]**, **[:graph:]**, **[:lower:]**, **[:print:]**, **[:punct:]**, **[:space:]**, **[:upper:]**, and **[:xdigit:]**. For example, **[[[:alnum:]]]** means the character class of numbers and letters in the current locale. In the C locale and ASCII character set encoding, this is the same as **[0-9A-Za-z]**. (Note that the brackets in these class names are part of the symbolic names, and must be included in addition to the brackets delimiting the bracket expression.) Most meta-characters lose their special meaning inside bracket expressions. To include a literal] place it first in the list. Similarly, to include a literal ^ place it anywhere but first. Finally, to include a literal – place it last.

Anchoring

The caret ^ and the dollar sign \$ are meta-characters that respectively match the empty string at the beginning and end of a line.

The Backslash Character and Special Expressions

The symbols \< and \> respectively match the empty string at the beginning and end of a word. The symbol \b matches the empty string at the edge of a word, and \B matches the empty string provided it's *not* at the edge of a word. The symbol \w is a synonym for [_[:alnum:]] and \W is a synonym for [^_[:alnum:]].

Repetition

A regular expression may be followed by one of several repetition operators:

- ? The preceding item is optional and matched at most once.
- * The preceding item will be matched zero or more times.
- + The preceding item will be matched one or more times.
- {n} The preceding item is matched exactly *n* times.
- {n,} The preceding item is matched *n* or more times.
- {,m} The preceding item is matched at most *m* times. This is a GNU extension.
- {n,m} The preceding item is matched at least *n* times, but not more than *m* times.

Concatenation

Two regular expressions may be concatenated; the resulting regular expression matches any string formed by concatenating two substrings that respectively match the concatenated expressions.

Alternation

Two regular expressions may be joined by the infix operator |; the resulting regular expression matches any string matching either alternate expression.

Precedence

Repetition takes precedence over concatenation, which in turn takes precedence over alternation. A whole expression may be enclosed in parentheses to override these precedence rules and form a subexpression.

Back-references and Subexpressions

The back-reference *n*, where *n* is a single digit, matches the substring previously matched by the *n*th parenthesized subexpression of the regular expression.

Basic vs Extended Regular Expressions

In basic regular expressions the meta-characters ?, +, {, |, (, and) lose their special meaning; instead use the backslashed versions \?, \+, \{, \|, \(), and \).

EXIT STATUS

Normally the exit status is 0 if a line is selected, 1 if no lines were selected, and 2 if an error occurred. However, if the **-q** or **--quiet** or **--silent** is used and a line is selected, the exit status is 0 even if an error occurred.

ENVIRONMENT

The behavior of **grep** is affected by the following environment variables.

The locale for category **LC_foo** is specified by examining the three environment variables **LC_ALL**, **LC_foo**, **LANG**, in that order. The first of these variables that is set specifies the locale. For example, if **LC_ALL** is not set, but **LC_MESSAGES** is set to **pt_BR**, then the Brazilian Portuguese locale is used for the **LC_MESSAGES** category. The C locale is used if none of these environment variables are set, if the locale catalog is not installed, or if **grep** was not compiled with national language support (NLS). The shell command **locale -a** lists locales that are currently available.

GREP_COLOR

This variable specifies the color used to highlight matched (non-empty) text. It is deprecated in favor of **GREP_COLORS**, but still supported. The **mt**, **ms**, and **mc** capabilities of **GREP_COLORS** have priority over it. It can only specify the color used to highlight the matching non-empty text in any matching line (a selected line when the **-v** command-line option is omitted, or a context line when **-v** is specified). The default is **01;31**, which means a bold red foreground text on the terminal's default background.

GREP_COLORS

Specifies the colors and other attributes used to highlight various parts of the output. Its value is a colon-separated list of capabilities that defaults to **ms=01;31;mc=01;31;sl=:cx=:fn=35:ln=32:bn=32:se=36** with the **rv** and **ne** boolean capabilities omitted (i.e., false). Supported capabilities are as follows.

sl= SGR substring for whole selected lines (i.e., matching lines when the **-v** command-line option is omitted, or non-matching lines when **-v** is specified). If however the boolean **rv** capability and the **-v** command-line option are both specified, it applies to context matching lines instead. The default is empty (i.e., the terminal's default color pair).

cx= SGR substring for whole context lines (i.e., non-matching lines when the **-v** command-line option is omitted, or matching lines when **-v** is specified). If however the boolean **rv** capability and the **-v** command-line option are both specified, it applies to selected non-matching lines instead. The default is empty (i.e., the terminal's default color pair).

rv Boolean value that reverses (swaps) the meanings of the **sl=** and **cx=** capabilities when the **-v** command-line option is specified. The default is false (i.e., the capability is omitted).

mt=01;31

SGR substring for matching non-empty text in any matching line (i.e., a selected line when the **-v** command-line option is omitted, or a context line when **-v** is specified). Setting this is equivalent to setting both **ms=** and **mc=** at once to the same value. The default is a bold red text foreground over the current line background.

ms=01;31

SGR substring for matching non-empty text in a selected line. (This is only used when the **-v** command-line option is omitted.) The effect of the **sl=** (or **cx=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

mc=01;31

SGR substring for matching non-empty text in a context line. (This is only used when the **-v** command-line option is specified.) The effect of the **cx=** (or **sl=** if **rv**) capability remains active when this kicks in. The default is a bold red text foreground over the current line background.

fn=35 SGR substring for file names prefixing any content line. The default is a magenta text foreground over the terminal's default background.

ln=32 SGR substring for line numbers prefixing any content line. The default is a green text foreground over the terminal's default background.

bn=32 SGR substring for byte offsets prefixing any content line. The default is a green text foreground over the terminal's default background.

se=36 SGR substring for separators that are inserted between selected line fields (:), between context line fields (-), and between groups of adjacent lines when nonzero context is specified (--). The default is a cyan text foreground over the terminal's default background.

ne Boolean value that prevents clearing to the end of line using Erase in Line (EL) to Right (\33[K) each time a colorized item ends. This is needed on terminals on which EL is not supported. It is otherwise useful on terminals for which the **back_color_erase** (**bce**) boolean terminfo capability does not apply, when the chosen highlight colors do not affect the background, or when EL is too slow or causes too much flicker. The default is false (i.e., the capability is omitted).

Note that boolean capabilities have no =... part. They are omitted (i.e., false) by default and become true when specified.

See the Select Graphic Rendition (SGR) section in the documentation of the text terminal that is used for permitted values and their meaning as character attributes. These substring values are integers in decimal representation and can be concatenated with semicolons. **grep** takes care of assembling the result into a complete SGR sequence (**\33[...m**). Common values to concatenate include **1** for bold, **4** for underline, **5** for blink, **7** for inverse, **39** for default foreground color, **30** to **37** for foreground colors, **90** to **97** for 16-color mode foreground colors, **38;5;0** to **38;5;255** for 88-color and 256-color modes foreground colors, **49** for default background color, **40** to **47** for background colors, **100** to **107** for 16-color mode background colors, and **48;5;0** to **48;5;255** for 88-color and 256-color modes background colors.

LC_ALL, LC_COLLATE, LANG

These variables specify the locale for the **LC_COLLATE** category, which determines the collating sequence used to interpret range expressions like **[a-z]**.

LC_ALL, LC_CTYPE, LANG

These variables specify the locale for the **LC_CTYPE** category, which determines the type of characters, e.g., which characters are whitespace. This category also determines the character encoding, that is, whether text is encoded in UTF-8, ASCII, or some other encoding. In the C or POSIX locale, all characters are encoded as a single byte and every byte is a valid character.

LC_ALL, LC_MESSAGES, LANG

These variables specify the locale for the **LC_MESSAGES** category, which determines the language that **grep** uses for messages. The default C locale uses American English messages.

POSIXLY_CORRECT

If set, **grep** behaves as POSIX requires; otherwise, **grep** behaves more like other GNU programs. POSIX requires that options that follow file names must be treated as file names; by default, such options are permuted to the front of the operand list and are treated as options. Also, POSIX requires that unrecognized options be diagnosed as “illegal”, but since they are not really against the law the default is to diagnose them as “invalid”. **POSIXLY_CORRECT** also disables **_N_GNU_nonoption_argv_flags_**, described below.

_N_GNU_nonoption_argv_flags_

(Here *N* is **grep**’s numeric process ID.) If the *i*th character of this environment variable’s value is **1**, do not consider the *i*th operand of **grep** to be an option, even if it appears to be one. A shell can put this variable in the environment for each command it runs, specifying which operands are the results of file name wildcard expansion and therefore should not be treated as options. This behavior is available only with the GNU C library, and only when **POSIXLY_CORRECT** is not set.

NOTES

This man page is maintained only fitfully; the full documentation is often more up-to-date.

COPYRIGHT

Copyright 1998-2000, 2002, 2005-2021 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

BUGS

Reporting Bugs

Email bug reports to the bug-reporting address [<bug-grep@gnu.org>](mailto:bug-grep@gnu.org). An email archive [<https://lists.gnu.org/mailman/listinfo/bug-grep>](https://lists.gnu.org/mailman/listinfo/bug-grep) and a bug tracker [<https://debbugs.gnu.org/cgi/pkgreport.cgi?package=grep>](https://debbugs.gnu.org/cgi/pkgreport.cgi?package=grep) are available.

Known Bugs

Large repetition counts in the **{n,m}** construct may cause **grep** to use lots of memory. In addition, certain other obscure regular expressions require exponential time and space, and may cause **grep** to run out of memory.

Back-references are very slow, and may require exponential time.

EXAMPLE

The following example outputs the location and contents of any line containing “f” and ending in “.c”, within all files in the current directory whose names contain “g” and end in “.h”. The **-n** option outputs line numbers, the **--** argument treats expansions of “*g*.h” starting with “-” as file names not options, and the empty file /dev/null causes file names to be output even if only one file name happens to be of the form “*g*.h”.

```
$ grep -n -- 'f.*\.c$' *g*.h /dev/null
argmatch.h:1:/* definitions and prototypes for argmatch.c
```

The only line that matches is line 1 of argmatch.h. Note that the regular expression syntax used in the pattern differs from the globbing syntax that the shell uses to match file names.

SEE ALSO

Regular Manual Pages

awk(1), **cmp(1)**, **diff(1)**, **find(1)**, **perl(1)**, **sed(1)**, **sort(1)**, **xargs(1)**, **read(2)**, **pcre(3)**, **pcresyntax(3)**, **pcrepattern(3)**, **terminfo(5)**, **glob(7)**, **regex(7)**

Full Documentation

A complete manual <<https://www.gnu.org/software/grep/manual/>> is available. If the **info** and **grep** programs are properly installed at your site, the command

```
info grep
```

should give you access to the complete manual.

NAME

group – user group file

DESCRIPTION

The */etc/group* file is a text file that defines the groups on the system. There is one entry per line, with the following format:

group_name:*password*:*GID*:*user_list*

The fields are as follows:

group_name

the name of the group.

password

the (encrypted) group password. If this field is empty, no password is needed.

GID

the numeric group ID.

user_list

a list of the usernames that are members of this group, separated by commas.

FILES

/etc/group

BUGS

As the 4.2BSD **initgroups**(3) man page says: no one seems to keep */etc/group* up-to-date.

SEE ALSO

chgrp(1), **gpasswd(1)**, **groups(1)**, **login(1)**, **newgrp(1)**, **sg(1)**, **getgrent(3)**, **getgrnam(3)**, **gshadow(5)**, **passwd(5)**, **vigr(8)**

NAME

`group.conf` – configuration file for the pam_group module

DESCRIPTION

The pam_group PAM module does not authenticate the user, but instead it grants group memberships (in the credential setting phase of the authentication module) to the user. Such memberships are based on the service they are applying for.

For this module to function correctly there must be a correctly formatted /etc/security/group.conf file present. White spaces are ignored and lines maybe extended with '\' (escaped newlines). Text following a '#' is ignored to the end of the line.

The syntax of the lines is as follows:

`services;ttys;users;times;groups`

The first field, the *services* field, is a logic list of PAM service names that the rule applies to.

The second field, the *tty* field, is a logic list of terminal names that this rule applies to.

The third field, the *users* field, is a logic list of users, or a UNIX group, or a netgroup of users to whom this rule applies. Group names are preceded by a '%' symbol, while netgroup names are preceded by a '@' symbol.

A logic list namely means individual tokens that are optionally prefixed with '!' (logical not) and separated with '&' (logical and) and '||' (logical or).

For these items the simple wildcard '*' may be used only once. With UNIX groups or netgroups no wildcards or logic operators are allowed.

The *times* field is used to indicate "when" these groups are to be given to the user. The format here is a logic list of day/time-range entries. The days are specified by a sequence of two character entries, MoTuSa for example is Monday Tuesday and Saturday. Note that repeated days are unset MoMo = no day, and MoWk = all weekdays bar Monday. The two character combinations accepted are Mo Tu We Th Fr Sa Su Wk Wd Al, the last two being week-end days and all 7 days of the week respectively. As a final example, AlFr means all days except Friday.

Each day/time-range can be prefixed with a '!' to indicate "anything but". The time-range part is two 24-hour times HHMM, separated by a hyphen, indicating the start and finish time (if the finish time is smaller than the start time it is deemed to apply on the following day).

The *groups* field is a comma or space separated list of groups that the user inherits membership of. These groups are added if the previous fields are satisfied by the user's request.

For a rule to be active, ALL of service+ttys+users must be satisfied by the applying process.

EXAMPLES

These are some example lines which might be specified in /etc/security/group.conf.

Running 'xsh' on tty* (any ttyXXX device), the user 'us' is given access to the floppy (through membership of the floppy group)

```
xsh;tty*&!ttyp*;us;Al0000-2400;floppy
```

Running 'xsh' on tty* (any ttyXXX device), the users 'sword', 'pike' and 'shield' are given access to games (through membership of the floppy group) after work hours.

```
xsh; tty* ;sword|pike|shield;!Wk0900-1800;games, sound
xsh; tty* ;*;Al0900-1800;floppy
```

Any member of the group 'admin' running 'xsh' on tty*, is granted access (at any time) to the group 'plugdev'

```
xsh; tty* ;%admin;Al0000-2400;plugdev
```

SEE ALSO

pam_group(8), pam.d(5), pam(7)

AUTHOR

pam_group was written by Andrew G. Morgan <morgan@kernel.org>.

NAME

groupadd – create a new group

SYNOPSIS

groupadd [*options*] *group*

DESCRIPTION

The **groupadd** command creates a new group account using the values specified on the command line plus the default values from the system. The new group will be entered into the system files as needed.

OPTIONS

The options which apply to the **groupadd** command are:

-f, --force

This option causes the command to simply exit with success status if the specified group already exists. When used with **-g**, and the specified GID already exists, another (unique) GID is chosen (i.e. **-g** is turned off).

-g, --gid *GID*

The numerical value of the group's ID. This value must be unique, unless the **-o** option is used. The value must be non-negative. The default is to use the smallest ID value greater than or equal to **GID_MIN** and greater than every other group.

See also the **-r** option and the **GID_MAX** description.

-h, --help

Display help message and exit.

-K, --key *KEY=VALUE*

Overrides /etc/login.defs defaults (**GID_MIN**, **GID_MAX** and others). Multiple **-K** options can be specified.

Example: **-K GID_MIN=100 -K GID_MAX=499**

Note: **-K GID_MIN=10,GID_MAX=499** doesn't work yet.

-o, --non-unique

This option permits to add a group with a non-unique GID.

-p, --password *PASSWORD*

The encrypted password, as returned by **crypt(3)**. The default is to disable the password.

Note: This option is not recommended because the password (or encrypted password) will be visible by users listing the processes.

You should make sure the password respects the system's password policy.

-r, --system

Create a system group.

The numeric identifiers of new system groups are chosen in the **SYS_GID_MIN**–**SYS_GID_MAX** range, defined in login.defs, instead of **GID_MIN**–**GID_MAX**.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix *PREFIX_DIR*

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

GID_MAX (number), **GID_MIN** (number)

Range of group IDs used for the creation of regular groups by **useradd**, **groupadd**, or **newusers**.

The default value for **GID_MIN** (resp. **GID_MAX**) is 1000 (resp. 60000).

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

SYS_GID_MAX (number), **SYS_GID_MIN** (number)

Range of group IDs used for the creation of system groups by **useradd**, **groupadd**, or **newusers**.

The default value for **SYS_GID_MIN** (resp. **SYS_GID_MAX**) is 101 (resp. **GID_MIN**-1).

FILES

/etc/group

Group account information.

/etc/gshadow

Secure group account information.

/etc/login.defs

Shadow password suite configuration.

CAVEATS

It is usually recommended to only use groupnames that begin with a lower case letter or an underscore, followed by lower case letters, digits, underscores, or dashes. They can end with a dollar sign. In regular expression terms: [a-z_][a-z0-9_-]*[\$]?

On Debian, the only constraints are that groupnames must neither start with a dash ('-') nor plus ('+') nor tilde ('~') nor contain a colon (':'), a comma (','), or a whitespace (space: ' ', end of line: '\n', tabulation: '\t', etc.).

On Ubuntu, the same constraints as Debian are in place, with the additional constraint that the groupname cannot be fully numeric. This includes octal and hexadecimal syntax.

Groupnames may only be up to 32 characters long.

You may not add a NIS or LDAP group. This must be performed on the corresponding server.

If the groupname already exists in an external group database such as NIS or LDAP, **groupadd** will deny the group creation request.

EXIT VALUES

The **groupadd** command exits with the following values:

0

success

2

invalid command syntax

3

invalid argument to option

4

GID not unique (when **-o** not used)

9

group name not unique

10

can't update group file

SEE ALSO

chfn(1), chsh(1), passwd(1), gpasswd(8), groupdel(8), groupmod(8), login.defs(5), useradd(8), userdel(8), usermod(8).

NAME

groupdel – delete a group

SYNOPSIS

groupdel [*options*] *GROUP*

DESCRIPTION

The **groupdel** command modifies the system account files, deleting all entries that refer to *GROUP*. The named group must exist.

OPTIONS

The options which apply to the **groupdel** command are:

-h, --help

Display help message and exit.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix *PREFIX_DIR*

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

CAVEATS

You may not remove the primary group of any existing user. You must remove the user before you remove the group.

You should manually check all file systems to ensure that no files remain owned by this group.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

FILES

/etc/group

Group account information.

/etc/gshadow

Secure group account information.

EXIT VALUES

The **groupdel** command exits with the following values:

0

success

2

invalid command syntax

6
specified group doesn't exist

8
can't remove user's primary group

10
can't update group file

SEE ALSO

chfn(1), chsh(1), passwd(1), gpasswd(8), groupadd(8), groupmod(8), useradd(8), userdel(8), usermod(8).

NAME

groupmod – modify a group definition on the system

SYNOPSIS

groupmod [*options*] *GROUP*

DESCRIPTION

The **groupmod** command modifies the definition of the specified *GROUP* by modifying the appropriate entry in the group database.

OPTIONS

The options which apply to the **groupmod** command are:

-g, --gid *GID*

The group ID of the given *GROUP* will be changed to *GID*.

The value of *GID* must be a non-negative decimal integer. This value must be unique, unless the **-o** option is used.

Users who use the group as primary group will be updated to keep the group as their primary group.

Any files that have the old group ID and must continue to belong to *GROUP*, must have their group ID changed manually.

No checks will be performed with regard to the **GID_MIN**, **GID_MAX**, **SYS_GID_MIN**, or **SYS_GID_MAX** from /etc/login.defs.

-h, --help

Display help message and exit.

-n, --new-name *NEW_GROUP*

The name of the group will be changed from *GROUP* to *NEW_GROUP* name.

-o, --non-unique

When used with the **-g** option, allow to change the group *GID* to a non-unique value.

-p, --password *PASSWORD*

The encrypted password, as returned by **crypt(3)**.

Note: This option is not recommended because the password (or encrypted password) will be visible by users listing the processes.

You should make sure the password respects the system's password policy.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix *PREFIX_DIR*

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

FILES

/etc/group	Group account information.
/etc/gshadow	Secure group account information.
/etc/login.defs	Shadow password suite configuration.
/etc/passwd	User account information.

EXIT VALUES

The **groupmod** command exits with the following values:

<i>0</i>	E_SUCCESS: success
<i>2</i>	E_USAGE: invalid command syntax
<i>3</i>	E_BAD_ARG: invalid argument to option
<i>4</i>	E_GID_IN_USE: specified group doesn't exist
<i>6</i>	E_NOTFOUND: specified group doesn't exist
<i>9</i>	E_NAME_IN_USE: group name already in use
<i>10</i>	E_GRP_UPDATE: can't update group file
<i>11</i>	E_CLEANUP_SERVICE: can't setup cleanup service
<i>12</i>	E_PAM_USERNAME: can't determine your username for use with pam
<i>13</i>	E_PAM_ERROR: pam returned an error, see syslog facility id groupmod for the PAM error message

SEE ALSO

chfn(1), **chsh(1)**, **passwd(1)**, **gpasswd(8)**, **groupadd(8)**, **groupdel(8)**, **login.defs(5)**, **useradd(8)**, **userdel(8)**, **usermod(8)**.

NAME

groups – print the groups a user is in

SYNOPSIS

groups [*OPTION*]... [*USERNAME*]...

DESCRIPTION

Print group memberships for each *USERNAME* or, if no *USERNAME* is specified, for the current process (which may differ if the groups database has changed).

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by David MacKenzie and James Youngman.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

getent(1)

Full documentation <<https://www.gnu.org/software/coreutils/groups>>
or available locally via: info '(coreutils) groups invocation'

NAME

grub-install – install GRUB to a device

SYNOPSIS

grub-install [OPTION...] [OPTION] [INSTALL_DEVICE]

DESCRIPTION

Install GRUB on your drive.

--compress=no|xz|gz|lzo
compress GRUB files [optional]

--disable-shim-lock
disable shim_lock verifier

--dtb=FILE
embed a specific DTB

-d, --directory=DIR
use images and modules under DIR [default=/usr/lib/grub/<platform>]

--fonts=FONTS
install FONTS [default=unicode]

--install-modules=MODULES
install only MODULES and their dependencies [default=all]

-k, --pubkey=FILE
embed FILE as public key for signature checking

--locale-directory=DIR use translations under DIR
[default=/usr/share/locale]

--locales=LOCALES
install only LOCALES [default=all]

--modules=MODULES
pre-load specified modules MODULES

--sbat=FILE
SBAT metadata

--themes=THEMES
install THEMES [default=starfield]

-v, --verbose
print verbose messages.

--allow-floppy
make the drive also bootable as floppy (default for fdX devices). May break on some BIOSes.

--boot-directory=DIR
install GRUB images under the directory DIR/grub instead of the boot/grub directory

--bootloader-id=ID
the ID of bootloader. This option is only available on EFI and Macs.

--core-compress=xz|none|auto
choose the compression to use for core image

--disk-module=MODULE
disk module to use (biosdisk or native). This option is only available on BIOS target.

--efi-directory=DIR
use DIR as the EFI System Partition root.

--force
install even if problems are detected

--force-file-id
use identifier file even if UUID is available

--label-bgcolor=COLOR
use COLOR for label background

--label-color=COLOR
use COLOR for label

--label-font=FILE
use FILE as font for label

--macppc-directory=DIR use DIR for PPC MAC install.

--no-bootsector
do not install bootsector

--no-extra-removable
Do not install bootloader code to the removable media path. This option is only available on EFI.

--no-nvram
don't update the 'boot-device'/'Boot*' NVRAM variables. This option is only available on EFI and IEEE1275 targets.

--no-rs-codes
Do not apply any reed-solomon codes when embedding core.img. This option is only available on x86 BIOS targets.

--no-uefi-secure-boot
do not install an image usable with UEFI Secure Boot, even if the system was currently started using it. This option is only available on EFI.

--product-version=STRING
use STRING as product version

--recheck
delete device map if it already exists

--removable
the installation device is removable. This option is only available on EFI.

-s, --skip-fs-probe
do not probe for filesystems in DEVICE

--target=TARGET
install GRUB for TARGET platform [default=i386-pc]; available targets: arm-coreboot, arm-efi, arm-uboot, arm64-efi, i386-coreboot, i386-efi, i386-ieee1275, i386-multiboot, i386-pc, i386-qemu, i386-xen, i386-xen_pv, ia64-efi, mips-arc, mips-qemu_mips, mipsel-arc, mipsel-loongson, mipsel-qemu_mips, powerpc-ieee1275, riscv32-efi, riscv64-efi, sparc64-ieee1275, x86_64-efi, x86_64-xen

--uefi-secure-boot
install an image usable with UEFI Secure Boot. This option is only available on EFI and if the grub-efi-amd64-signed package is installed.

-?, --help
give this help list

--usage
give a short usage message

-V, --version
print program version

Mandatory or optional arguments to long options are also mandatory or optional for any corresponding short options.

INSTALL_DEVICE must be system device filename. grub-install copies GRUB images into boot/grub. On some platforms, it may also install GRUB into the boot sector.

REPORTING BUGS

Report bugs to <bug-grub@gnu.org>.

SEE ALSO

grub-mkconfig(8), grub-mkimage(1), grub-mkrescue(1)

The full documentation for **grub-install** is maintained as a Texinfo manual. If the **info** and **grub-install** programs are properly installed at your site, the command

info grub-install

should give you access to the complete manual.

NAME

grub-mkconfig – generate a GRUB configuration file

SYNOPSIS

grub-mkconfig [*OPTION*]

DESCRIPTION

Generate a grub config file

-o, --output=FILE

output generated config to FILE [default=stdout]

-h, --help

print this message and exit

-V, --version

print the version information and exit

REPORTING BUGS

Report bugs to <bug-grub@gnu.org>.

SEE ALSO

grub-install(8)

The full documentation for **grub-mkconfig** is maintained as a Texinfo manual. If the **info** and **grub-mkconfig** programs are properly installed at your site, the command

info grub-mkconfig

should give you access to the complete manual.

NAME

gzip, *gunzip*, *zcat* – compress or expand files

SYNOPSIS

```
gzip [ -acdfhklLnNrtvV19 ] [-S suffix] [ name ... ]
gunzip [ -acfhlkLlnNrtvV ] [-S suffix] [ name ... ]
zcat [ -fhLV ] [ name ... ]
```

DESCRIPTION

Gzip reduces the size of the named files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension **.gz**, while keeping the same ownership modes, access and modification times. (The default extension is **z** for MSDOS, OS/2 FAT, Windows NT FAT and Atari.) If no files are specified, or if a file name is **"."**, the standard input is compressed to the standard output. *Gzip* will only attempt to compress regular files. In particular, it will ignore symbolic links.

If the compressed file name is too long for its file system, *gzip* truncates it. *Gzip* attempts to truncate only the parts of the file name longer than 3 characters. (A part is delimited by dots.) If the name consists of small parts only, the longest parts are truncated. For example, if file names are limited to 14 characters, *gzip.msdos.exe* is compressed to *gzi.msd.exe.gz*. Names are not truncated on systems which do not have a limit on file name length.

By default, *gzip* keeps the original file name and timestamp in the compressed file. These are used when decompressing the file with the **-N** option. This is useful when the compressed file name was truncated or when the timestamp was not preserved after a file transfer.

Compressed files can be restored to their original form using *gzip -d* or *gunzip* or *zcat*. If the original name saved in the compressed file is not suitable for its file system, a new name is constructed from the original one to make it legal.

gunzip takes a list of files on its command line and replaces each file whose name ends with **.gz**, **-gz**, **.z**, **-z**, or **_z** (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension. *gunzip* also recognizes the special extensions **.tgz** and **.taz** as shorthands for **.tar.gz** and **.tar.Z** respectively. When compressing, *gzip* uses the **.tgz** extension if necessary instead of truncating a file with a **.tar** extension.

gunzip can currently decompress files created by *gzip*, *zip*, *compress*, *compress -H* or *pack*. The detection of the input format is automatic. When using the first two formats, *gunzip* checks a 32 bit CRC. For *pack* and *gunzip* checks the uncompressed length. The standard *compress* format was not designed to allow consistency checks. However *gunzip* is sometimes able to detect a bad **.Z** file. If you get an error when uncompressing a **.Z** file, do not assume that the **.Z** file is correct simply because the standard *uncompress* does not complain. This generally means that the standard *uncompress* does not check its input, and happily generates garbage output. The SCO compress **-H** format (lzh compression method) does not include a CRC but also allows some consistency checks.

Files created by *zip* can be uncompressed by *gzip* only if they have a single member compressed with the 'deflation' method. This feature is only intended to help conversion of tar.zip files to the tar.gz format. To extract a *zip* file with a single member, use a command like *gunzip <foo.zip* or *gunzip -S .zip foo.zip*. To extract zip files with several members, use *unzip* instead of *gunzip*.

zcat is identical to *gunzip -c*. (On some systems, *zcat* may be installed as *gzcat* to preserve the original link to *compress*.) *zcat* uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output. *zcat* will uncompress files that have the correct magic number whether they have a **.gz** suffix or not.

Gzip uses the Lempel-Ziv algorithm used in *zip* and PKZIP. The amount of compression obtained depends on the size of the input and the distribution of common substrings. Typically, text such as source code or English is reduced by 60–70%. Compression is generally much better than that achieved by LZW (as used in *compress*), Huffman coding (as used in *pack*), or adaptive Huffman coding (*compact*).

Compression is always performed, even if the compressed file is slightly larger than the original. The worst case expansion is a few bytes for the *gzip* file header, plus 5 bytes every 32K block, or an expansion ratio of

0.015% for large files. Note that the actual number of used disk blocks almost never increases. *gzip* preserves the mode, ownership and timestamps of files when compressing or decompressing.

OPTIONS

-a --ascii

Ascii text mode: convert end-of-lines using local conventions. This option is supported only on some non-Unix systems. For MSDOS, CR LF is converted to LF when compressing, and LF is converted to CR LF when decompressing.

-c --stdout --to-stdout

Write output on standard output; keep original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d --decompress --uncompress

Decompress.

-f --force

Force compression or decompression even if the file has multiple links or the corresponding file already exists, or if the compressed data is read from or written to a terminal. If the input data is not in a format recognized by *gzip*, and if the option **--stdout** is also given, copy the input data without change to the standard output: let *zcat* behave as *cat*. If **-f** is not given, and when not running in the background, *gzip* prompts to verify whether an existing file should be overwritten.

-h --help

Display a help screen and quit.

-k --keep

Keep (don't delete) input files during compression or decompression.

-l --list For each compressed file, list the following fields:

- compressed size: size of the compressed file
- uncompressed size: size of the uncompressed file
- ratio: compression ratio (0.0% if unknown)
- uncompressed_name: name of the uncompressed file

The uncompressed size is given as -1 for files not in gzip format, such as compressed .Z files. To get the uncompressed size for such a file, you can use:

```
zcat file.Z | wc -c
```

In combination with the **--verbose** option, the following fields are also displayed:

- method: compression method
- crc: the 32-bit CRC of the uncompressed data
- date & time: timestamp for the uncompressed file

The compression methods currently supported are deflate, compress, lzh (SCO compress -H) and pack. The crc is given as fffffff for a file not in gzip format.

With **--name**, the uncompressed name, date and time are those stored within the compress file if present.

With **--verbose**, the size totals and compression ratio for all files is also displayed, unless some sizes are unknown. With **--quiet**, the title and totals lines are not displayed.

-L --license

Display the *gzip* license and quit.

-n --no-name

When compressing, do not save the original file name and timestamp by default. (The original name is always saved if the name had to be truncated.) When decompressing, do not restore the original file name if present (remove only the *gzip* suffix from the compressed file name) and do not restore the original timestamp if present (copy it from the compressed file). This option is the default when decompressing.

-N --name

When compressing, always save the original file name and timestamp; this is the default. When decompressing, restore the original file name and timestamp if present. This option is useful on systems which have a limit on file name length or when the timestamp has been lost after a file transfer.

-q --quiet

Suppress all warnings.

-r --recursive

Travel the directory structure recursively. If any of the file names specified on the command line are directories, *gzip* will descend into the directory and compress all the files it finds there (or decompress them in the case of *gunzip*).

-S .suf --suffix .suf

When compressing, use suffix *.suf* instead of *.gz*. Any non-empty suffix can be given, but suffixes other than *.z* and *.gz* should be avoided to avoid confusion when files are transferred to other systems.

When decompressing, add *.suf* to the beginning of the list of suffixes to try, when deriving an output file name from an input file name.

--synchronous

Use synchronous output. With this option, *gzip* is less likely to lose data during a system crash, but it can be considerably slower.

-t --test

Test. Check the compressed file integrity.

-v --verbose

Verbose. Display the name and percentage reduction for each file compressed or decompressed.

-V --version

Version. Display the version number and compilation options then quit.

-# --fast --best

Regulate the speed of compression using the specified digit *#*, where **-1** or **--fast** indicates the fastest compression method (less compression) and **-9** or **--best** indicates the slowest compression method (best compression). The default compression level is **-6** (that is, biased towards high compression at expense of speed).

--rsyncable

When you synchronize a compressed file between two computers, this option allows rsync to transfer only files that were changed in the archive instead of the entire archive. Normally, after a change is made to any file in the archive, the compression algorithm can generate a new version of the archive that does not match the previous version of the archive. In this case, rsync transfers the entire new version of the archive to the remote computer. With this option, rsync can transfer only the changed files as well as a small amount of metadata that is required to update the archive structure in the area that was changed.

ADVANCED USAGE

Multiple compressed files can be concatenated. In this case, *gunzip* will extract all members at once. For example:

```
gzip -c file1 > foo.gz
gzip -c file2 >> foo.gz
```

Then

```
gunzip -c foo
```

is equivalent to

```
cat file1 file2
```

In case of damage to one member of a .gz file, other members can still be recovered (if the damaged member is removed). However, you can get better compression by compressing all members at once:

```
cat file1 file2 | gzip > foo.gz
```

compresses better than

```
gzip -c file1 file2 > foo.gz
```

If you want to recompress concatenated files to get better compression, do:

```
gzip -cd old.gz | gzip > new.gz
```

If a compressed file consists of several members, the uncompressed size and CRC reported by the --list option applies to the last member only. If you need the uncompressed size for all members, you can use:

```
gzip -cd file.gz | wc -c
```

If you wish to create a single archive file with multiple members so that members can later be extracted independently, use an archiver such as tar or zip. GNU tar supports the -z option to invoke gzip transparently. gzip is designed as a complement to tar, not as a replacement.

ENVIRONMENT

The obsolescent environment variable **GZIP** can hold a set of default options for *gzip*. These options are interpreted first and can be overwritten by explicit command line parameters. As this can cause problems when using scripts, this feature is supported only for options that are reasonably likely to not cause too much harm, and *gzip* warns if it is used. This feature will be removed in a future release of *gzip*.

You can use an alias or script instead. For example, if *gzip* is in the directory **/usr/bin** you can prepend **\$HOME/bin** to your **PATH** and create an executable script **\$HOME/bin/gzip** containing the following:

```
#!/bin/sh
export PATH=/usr/bin
exec gzip -9 "$@"
```

SEE ALSO

znew(1), *zcmp(1)*, *zmore(1)*, *zforce(1)*, *gzexe(1)*, *zip(1)*, *unzip(1)*, *compress(1)*

The *gzip* file format is specified in P. Deutsch, GZIP file format specification version 4.3, <<https://www.ietf.org/rfc/rfc1952.txt>>, Internet RFC 1952 (May 1996). The *zip* deflation format is specified in P. Deutsch, DEFLATE Compressed Data Format Specification version 1.3, <<https://www.ietf.org/rfc/rfc1951.txt>>, Internet RFC 1951 (May 1996).

DIAGNOSTICS

Exit status is normally 0; if an error occurs, exit status is 1. If a warning occurs, exit status is 2.

Usage: gzip [-cdfhklLnNrtvV19] [-S suffix] [file ...]

 Invalid options were specified on the command line.

file: not in gzip format

 The file specified to *gunzip* has not been compressed.

file: Corrupt input. Use zcat to recover some data.

 The compressed file has been damaged. The data up to the point of failure can be recovered using

```
zcat file > recover
```

file: compressed with *xx* bits, can only handle *yy* bits

File was compressed (using LZW) by a program that could deal with more *bits* than the decompress code on this machine. Recompress the file with gzip, which compresses better and uses less memory.

file: already has .gz suffix -- unchanged

 The file is assumed to be already compressed. Rename the file and try again.

file already exists; do you wish to overwrite (y or n)?

 Respond "y" if you want the output file to be replaced; "n" if not.

gunzip: corrupt input

 A SIGSEGV violation was detected which usually means that the input file has been corrupted.

xx.x% Percentage of the input saved by compression.

 (Relevant only for -v and -l.)

-- not a regular file or directory: ignored

 When the input file is not a regular file or directory, (e.g. a symbolic link, socket, FIFO, device file), it is left unaltered.

-- has *xx* other links: unchanged

 The input file has links; it is left unchanged. See *ln(1)* for more information. Use the -f flag to force compression of multiply-linked files.

CAVEATS

When writing compressed data to a tape, it is generally necessary to pad the output with zeroes up to a block boundary. When the data is read and the whole block is passed to *gunzip* for decompression, *gunzip* detects that there is extra trailing garbage after the compressed data and emits a warning by default. You can use the --quiet option to suppress the warning.

BUGS

The gzip format represents the input size modulo 2^32, so the --list option reports incorrect uncompressed sizes and compression ratios for uncompressed files 4 GB and larger. To work around this problem, you can use the following command to discover a large uncompressed file's true size:

```
zcat file.gz | wc -c
```

The --list option reports sizes as -1 and crc as ffffffff if the compressed file is on a non seekable media.

In some rare cases, the --best option gives worse compression than the default compression level (-6). On some highly redundant files, *compress* compresses better than *gzip*.

COPYRIGHT NOTICE

Copyright © 1998-1999, 2001-2002, 2012, 2015-2018 Free Software Foundation, Inc.

Copyright © 1992, 1993 Jean-loup Gailly

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Foundation.

NAME

gzip, *gunzip*, *zcat* – compress or expand files

SYNOPSIS

```
gzip [ -acdfhklLnNrtvV19 ] [-S suffix] [ name ... ]
gunzip [ -acfhlkLlnNrtvV ] [-S suffix] [ name ... ]
zcat [ -fhLV ] [ name ... ]
```

DESCRIPTION

Gzip reduces the size of the named files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension **.gz**, while keeping the same ownership modes, access and modification times. (The default extension is **z** for MSDOS, OS/2 FAT, Windows NT FAT and Atari.) If no files are specified, or if a file name is "**-**", the standard input is compressed to the standard output. *Gzip* will only attempt to compress regular files. In particular, it will ignore symbolic links.

If the compressed file name is too long for its file system, *gzip* truncates it. *Gzip* attempts to truncate only the parts of the file name longer than 3 characters. (A part is delimited by dots.) If the name consists of small parts only, the longest parts are truncated. For example, if file names are limited to 14 characters, *gzip.msdos.exe* is compressed to *gzi.msd.exe.gz*. Names are not truncated on systems which do not have a limit on file name length.

By default, *gzip* keeps the original file name and timestamp in the compressed file. These are used when decompressing the file with the **-N** option. This is useful when the compressed file name was truncated or when the timestamp was not preserved after a file transfer.

Compressed files can be restored to their original form using *gzip -d* or *gunzip* or *zcat*. If the original name saved in the compressed file is not suitable for its file system, a new name is constructed from the original one to make it legal.

gunzip takes a list of files on its command line and replaces each file whose name ends with **.gz**, **-gz**, **.z**, **-z**, or **_z** (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension. *gunzip* also recognizes the special extensions **.tgz** and **.taz** as shorthands for **.tar.gz** and **.tar.Z** respectively. When compressing, *gzip* uses the **.tgz** extension if necessary instead of truncating a file with a **.tar** extension.

gunzip can currently decompress files created by *gzip*, *zip*, *compress*, *compress -H* or *pack*. The detection of the input format is automatic. When using the first two formats, *gunzip* checks a 32 bit CRC. For *pack* and *gunzip* checks the uncompressed length. The standard *compress* format was not designed to allow consistency checks. However *gunzip* is sometimes able to detect a bad **.Z** file. If you get an error when uncompressing a **.Z** file, do not assume that the **.Z** file is correct simply because the standard *uncompress* does not complain. This generally means that the standard *uncompress* does not check its input, and happily generates garbage output. The SCO compress **-H** format (lzh compression method) does not include a CRC but also allows some consistency checks.

Files created by *zip* can be uncompressed by *gzip* only if they have a single member compressed with the 'deflation' method. This feature is only intended to help conversion of tar.zip files to the tar.gz format. To extract a *zip* file with a single member, use a command like *gunzip <foo.zip* or *gunzip -S .zip foo.zip*. To extract zip files with several members, use *unzip* instead of *gunzip*.

zcat is identical to *gunzip -c*. (On some systems, *zcat* may be installed as *gzcat* to preserve the original link to *compress*.) *zcat* uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output. *zcat* will uncompress files that have the correct magic number whether they have a **.gz** suffix or not.

Gzip uses the Lempel-Ziv algorithm used in *zip* and PKZIP. The amount of compression obtained depends on the size of the input and the distribution of common substrings. Typically, text such as source code or English is reduced by 60–70%. Compression is generally much better than that achieved by LZW (as used in *compress*), Huffman coding (as used in *pack*), or adaptive Huffman coding (*compact*).

Compression is always performed, even if the compressed file is slightly larger than the original. The worst case expansion is a few bytes for the *gzip* file header, plus 5 bytes every 32K block, or an expansion ratio of

0.015% for large files. Note that the actual number of used disk blocks almost never increases. *gzip* preserves the mode, ownership and timestamps of files when compressing or decompressing.

OPTIONS

-a --ascii

Ascii text mode: convert end-of-lines using local conventions. This option is supported only on some non-Unix systems. For MSDOS, CR LF is converted to LF when compressing, and LF is converted to CR LF when decompressing.

-c --stdout --to-stdout

Write output on standard output; keep original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d --decompress --uncompress

Decompress.

-f --force

Force compression or decompression even if the file has multiple links or the corresponding file already exists, or if the compressed data is read from or written to a terminal. If the input data is not in a format recognized by *gzip*, and if the option --stdout is also given, copy the input data without change to the standard output: let *zcat* behave as *cat*. If -f is not given, and when not running in the background, *gzip* prompts to verify whether an existing file should be overwritten.

-h --help

Display a help screen and quit.

-k --keep

Keep (don't delete) input files during compression or decompression.

-l --list For each compressed file, list the following fields:

compressed size: size of the compressed file
 uncompressed size: size of the uncompressed file
 ratio: compression ratio (0.0% if unknown)
 uncompressed_name: name of the uncompressed file

The uncompressed size is given as -1 for files not in gzip format, such as compressed .Z files. To get the uncompressed size for such a file, you can use:

```
zcat file.Z | wc -c
```

In combination with the --verbose option, the following fields are also displayed:

method: compression method
 crc: the 32-bit CRC of the uncompressed data
 date & time: timestamp for the uncompressed file

The compression methods currently supported are deflate, compress, lzh (SCO compress -H) and pack. The crc is given as fffffff for a file not in gzip format.

With --name, the uncompressed name, date and time are those stored within the compress file if present.

With --verbose, the size totals and compression ratio for all files is also displayed, unless some sizes are unknown. With --quiet, the title and totals lines are not displayed.

-L --license

Display the *gzip* license and quit.

-n --no-name

When compressing, do not save the original file name and timestamp by default. (The original name is always saved if the name had to be truncated.) When decompressing, do not restore the original file name if present (remove only the *gzip* suffix from the compressed file name) and do not restore the original timestamp if present (copy it from the compressed file). This option is the default when decompressing.

-N --name

When compressing, always save the original file name and timestamp; this is the default. When decompressing, restore the original file name and timestamp if present. This option is useful on systems which have a limit on file name length or when the timestamp has been lost after a file transfer.

-q --quiet

Suppress all warnings.

-r --recursive

Travel the directory structure recursively. If any of the file names specified on the command line are directories, *gzip* will descend into the directory and compress all the files it finds there (or decompress them in the case of *gunzip*).

-S .suf --suffix .suf

When compressing, use suffix *.suf* instead of *.gz*. Any non-empty suffix can be given, but suffixes other than *.z* and *.gz* should be avoided to avoid confusion when files are transferred to other systems.

When decompressing, add *.suf* to the beginning of the list of suffixes to try, when deriving an output file name from an input file name.

--synchronous

Use synchronous output. With this option, *gzip* is less likely to lose data during a system crash, but it can be considerably slower.

-t --test

Test. Check the compressed file integrity.

-v --verbose

Verbose. Display the name and percentage reduction for each file compressed or decompressed.

-V --version

Version. Display the version number and compilation options then quit.

-# --fast --best

Regulate the speed of compression using the specified digit *#*, where **-1** or **--fast** indicates the fastest compression method (less compression) and **-9** or **--best** indicates the slowest compression method (best compression). The default compression level is **-6** (that is, biased towards high compression at expense of speed).

--rsyncable

When you synchronize a compressed file between two computers, this option allows rsync to transfer only files that were changed in the archive instead of the entire archive. Normally, after a change is made to any file in the archive, the compression algorithm can generate a new version of the archive that does not match the previous version of the archive. In this case, rsync transfers the entire new version of the archive to the remote computer. With this option, rsync can transfer only the changed files as well as a small amount of metadata that is required to update the archive structure in the area that was changed.

ADVANCED USAGE

Multiple compressed files can be concatenated. In this case, *gunzip* will extract all members at once. For example:

```
gzip -c file1 > foo.gz
gzip -c file2 >> foo.gz
```

Then

```
gunzip -c foo
```

is equivalent to

```
cat file1 file2
```

In case of damage to one member of a .gz file, other members can still be recovered (if the damaged member is removed). However, you can get better compression by compressing all members at once:

```
cat file1 file2 | gzip > foo.gz
```

compresses better than

```
gzip -c file1 file2 > foo.gz
```

If you want to recompress concatenated files to get better compression, do:

```
gzip -cd old.gz | gzip > new.gz
```

If a compressed file consists of several members, the uncompressed size and CRC reported by the --list option applies to the last member only. If you need the uncompressed size for all members, you can use:

```
gzip -cd file.gz | wc -c
```

If you wish to create a single archive file with multiple members so that members can later be extracted independently, use an archiver such as tar or zip. GNU tar supports the -z option to invoke gzip transparently. gzip is designed as a complement to tar, not as a replacement.

ENVIRONMENT

The obsolescent environment variable **GZIP** can hold a set of default options for *gzip*. These options are interpreted first and can be overwritten by explicit command line parameters. As this can cause problems when using scripts, this feature is supported only for options that are reasonably likely to not cause too much harm, and *gzip* warns if it is used. This feature will be removed in a future release of *gzip*.

You can use an alias or script instead. For example, if *gzip* is in the directory **/usr/bin** you can prepend **\$HOME/bin** to your **PATH** and create an executable script **\$HOME/bin/gzip** containing the following:

```
#!/bin/sh
export PATH=/usr/bin
exec gzip -9 "$@"
```

SEE ALSO

znew(1), *zcmp(1)*, *zmore(1)*, *zforce(1)*, *gzexe(1)*, *zip(1)*, *unzip(1)*, *compress(1)*

The *gzip* file format is specified in P. Deutsch, GZIP file format specification version 4.3, <<https://www.ietf.org/rfc/rfc1952.txt>>, Internet RFC 1952 (May 1996). The *zip* deflation format is specified in P. Deutsch, DEFLATE Compressed Data Format Specification version 1.3, <<https://www.ietf.org/rfc/rfc1951.txt>>, Internet RFC 1951 (May 1996).

DIAGNOSTICS

Exit status is normally 0; if an error occurs, exit status is 1. If a warning occurs, exit status is 2.

Usage: gzip [-cdfhklLnNrtvV19] [-S suffix] [file ...]

Invalid options were specified on the command line.

file: not in gzip format

The file specified to *gunzip* has not been compressed.

file: Corrupt input. Use zcat to recover some data.

The compressed file has been damaged. The data up to the point of failure can be recovered using

```
zcat file > recover
```

file: compressed with xx bits, can only handle yy bits

File was compressed (using LZW) by a program that could deal with more *bits* than the decompress code on this machine. Recompress the file with gzip, which compresses better and uses less memory.

file: already has .gz suffix -- unchanged

The file is assumed to be already compressed. Rename the file and try again.

file already exists; do you wish to overwrite (y or n)?

Respond "y" if you want the output file to be replaced; "n" if not.

gunzip: corrupt input

A SIGSEGV violation was detected which usually means that the input file has been corrupted.

xx.x% Percentage of the input saved by compression.

(Relevant only for **-v** and **-l**.)

-- not a regular file or directory: ignored

When the input file is not a regular file or directory, (e.g. a symbolic link, socket, FIFO, device file), it is left unaltered.

-- has xx other links: unchanged

The input file has links; it is left unchanged. See *ln(1)* for more information. Use the **-f** flag to force compression of multiply-linked files.

CAVEATS

When writing compressed data to a tape, it is generally necessary to pad the output with zeroes up to a block boundary. When the data is read and the whole block is passed to *gunzip* for decompression, *gunzip* detects that there is extra trailing garbage after the compressed data and emits a warning by default. You can use the **--quiet** option to suppress the warning.

BUGS

The gzip format represents the input size modulo 2³², so the **--list** option reports incorrect uncompressed sizes and compression ratios for uncompressed files 4 GB and larger. To work around this problem, you can use the following command to discover a large uncompressed file's true size:

```
zcat file.gz | wc -c
```

The **--list** option reports sizes as -1 and crc as ffffffff if the compressed file is on a non seekable media.

In some rare cases, the **--best** option gives worse compression than the default compression level (-6). On some highly redundant files, *compress* compresses better than *gzip*.

COPYRIGHT NOTICE

Copyright © 1998-1999, 2001-2002, 2012, 2015-2018 Free Software Foundation, Inc.

Copyright © 1992, 1993 Jean-loup Gailly

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Foundation.

NAME

head – output the first part of files

SYNOPSIS

head [*OPTION*]... [*FILE*]...

DESCRIPTION

Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is **-**, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-c, --bytes=*JNUM*

print the first NUM bytes of each file; with the leading **'-**', print all but the last NUM bytes of each file

-n, --lines=*JNUM*

print the first NUM lines instead of the first 10; with the leading **'-**', print all but the last NUM lines of each file

-q, --quiet, --silent

never print headers giving file names

-v, --verbose

always print headers giving file names

-z, --zero-terminated

line delimiter is NUL, not newline

--help display this help and exit

--version

output version information and exit

NUM may have a multiplier suffix: b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y. Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

AUTHOR

Written by David MacKenzie and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

tail(1)

Full documentation <<https://www.gnu.org/software/coreutils/head>>
or available locally via: info '(coreutils) head invocation'

NAME

hostname – show or set the system's host name
domainname – show or set the system's NIS/YP domain name
ypdomainname – show or set the system's NIS/YP domain name
nisdomainname – show or set the system's NIS/YP domain name
dnsdomainname – show the system's DNS domain name

SYNOPSIS

```
hostname [-a|--alias] [-d|--domain] [-f|--fqdn|--long] [-A|--all-fqdns] [-i|--ip-address]
[-I|--all-ip-addresses] [-s|--short] [-y|--yp|--nis]
hostname [-b|--boot] [-F|--file filename] [hostname]
hostname [-h|--help] [-V|--version]

domainname [nisdomain] [-F file]
ypdomainname [nisdomain] [-F file]
nisdomainname [nisdomain] [-F file]

dnsdomainname
```

DESCRIPTION

Hostname is used to display the system's DNS name, and to display or set its hostname or NIS domain name.

GET NAME

When called without any arguments, the program displays the current names:

hostname will print the name of the system as returned by the **gethostname(2)** function.

domainname will print the NIS domainname of the system. **domainname** uses the **gethostname(2)** function, while **ypdomainname** and **nisdomainname** use the **getdomainname(2)**.

dnsdomainname will print the domain part of the FQDN (Fully Qualified Domain Name). The complete FQDN of the system is returned with **hostname --fqdn** (but see the warnings in section **THE FQDN** below).

SET NAME

When called with one argument or with the **--file** option, the commands set the host name or the NIS/YP domain name. **hostname** uses the **sethostname(2)** function, while all of the three **domainname**, **ypdomainname** and **nisdomainname** use **setdomainname(2)**. Note, that this is effective only until the next reboot. Edit /etc/hostname for permanent change.

Note, that only the super-user can change the names.

It is not possible to set the FQDN or the DNS domain name with the **dnsdomainname** command (see **THE FQDN** below).

The host name is usually set once at system startup in */etc/init.d/hostname.sh* (normally by reading the contents of a file which contains the host name, e.g. */etc/hostname*).

THE FQDN

The FQDN (Fully Qualified Domain Name) of the system is the name that the **resolver(3)** returns for the host name, such as, *ursula.example.com*. It is usually the hostname followed by the DNS domain name (the part after the first dot). You can check the FQDN using **hostname --fqdn** or the domain name using **dnsdomainname**.

You cannot change the FQDN with **hostname** or **dnsdomainname**.

The recommended method of setting the FQDN is to make the hostname be an alias for the fully qualified

name using */etc/hosts*, DNS, or NIS. For example, if the hostname was "ursula", one might have a line in */etc/hosts* which reads

```
127.0.1.1  ursula.example.com ursula
```

Technically: The FQDN is the name **getaddrinfo**(3) returns for the host name returned by **gethostname**(2). The DNS domain name is the part after the first dot.

Therefore it depends on the configuration of the resolver (usually in */etc/host.conf*) how you can change it. Usually the hosts file is parsed before DNS or NIS, so it is most common to change the FQDN in */etc/hosts*.

If a machine has multiple network interfaces/addresses or is used in a mobile environment, then it may either have multiple FQDNs/domain names or none at all. Therefore avoid using **hostname --fqdn**, **hostname --domain** and **dnsdomainname**. **hostname --ip-address** is subject to the same limitations so it should be avoided as well.

OPTIONS

-a, --alias

Display the alias name of the host (if used). This option is deprecated and should not be used anymore.

-A, --all-fqdns

Displays all FQDNs of the machine. This option enumerates all configured network addresses on all configured network interfaces, and translates them to DNS domain names. Addresses that cannot be translated (i.e. because they do not have an appropriate reverse IP entry) are skipped. Note that different addresses may resolve to the same name, therefore the output may contain duplicate entries. Do not make any assumptions about the order of the output.

-b, --boot

Always set a hostname; this allows the file specified by **-F** to be non-existent or empty, in which case the default hostname *localhost* will be used if none is yet set.

-d, --domain

Display the name of the DNS domain. Don't use the command **domainname** to get the DNS domain name because it will show the NIS domain name and not the DNS domain name. Use **dnsdomainname** instead. See the warnings in section **THE FQDN** above, and avoid using this option.

-f, --fqdn, --long

Display the FQDN (Fully Qualified Domain Name). A FQDN consists of a short host name and the DNS domain name. Unless you are using bind or NIS for host lookups you can change the FQDN and the DNS domain name (which is part of the FQDN) in the */etc/hosts* file. See the warnings in section **THE FQDN** above and use **hostname --all-fqdns** instead wherever possible.

-F, --file filename

Read the host name from the specified file. Comments (lines starting with a '#') are ignored.

-i, --ip-address

Display the network address(es) of the host name. Note that this works only if the host name can be resolved. Avoid using this option; use **hostname --all-ip-addresses** instead.

-I, --all-ip-addresses

Display all network addresses of the host. This option enumerates all configured addresses on all network interfaces. The loopback interface and IPv6 link-local addresses are omitted. Contrary to option **-i**, this option does not depend on name resolution. Do not make any assumptions about the order of the output.

-s, --short

Display the short host name. This is the host name cut at the first dot.

-V, --version

Print version information on standard output and exit successfully.

-y, --yp, --nis

Display the NIS domain name. If a parameter is given (or **--file name**) then root can also set a new NIS domain.

-h, --help

Print a usage message and exit.

NOTES

The address families **hostname** tries when looking up the FQDN, aliases and network addresses of the host are determined by the configuration of your resolver. For instance, on GNU Libc systems, the resolver can be instructed to try IPv6 lookups first by using the **inet6** option in **/etc/resolv.conf**.

FILES

/etc/hostname Historically this file was supposed to only contain the hostname and not the full canonical FQDN. Nowadays most software is able to cope with a full FQDN here. This file is read at boot time by the system initialization scripts to set the hostname.

/etc/hosts Usually, this is where one sets the domain name by aliasing the host name to the FQDN.

AUTHORS

Peter Tobias, <tobias@et-inf.fho-emden.de>

Bernd Eckenfels, <net-tools@lina.inka.de> (NIS and manpage).

Michael Meskes, <meskes@debian.org>

NAME

hostname – Local hostname configuration file

SYNOPSIS

/etc/hostname

DESCRIPTION

The /etc/hostname file configures the name of the local system. Unless overridden as described in the next section, **systemd**(1) will set this hostname during boot using the **sethostname**(2) system call.

The file should contain a single newline-terminated hostname string. Comments (lines starting with a "#") are ignored. The hostname should be composed of up to 64 7-bit ASCII lower-case alphanumeric characters or hyphens forming a valid DNS domain name. It is recommended that this name contains only a single label, i.e. without any dots. Invalid characters will be filtered out in an attempt to make the name valid, but obviously it is recommended to use a valid name and not rely on this filtering.

You may use **hostnamectl**(1) to change the value of this file during runtime from the command line. Use **systemd-firstboot**(1) to initialize it on mounted (but not booted) system images.

HOSTNAME SEMANTICS

systemd(1) and the associated tools will obtain the hostname in the following ways:

- If the kernel commandline parameter *systemd.hostname=* specifies a valid hostname, **systemd**(1) will use it to set the hostname during early boot, see **kernel-command-line**(7),
- Otherwise, the "static" hostname specified by /etc/hostname as described above will be used.
- Otherwise, a transient hostname may be set during runtime, for example based on information in a DHCP lease, see **systemd-hostnamed.service**(8). Both **NetworkManager**^[1] and **systemd-networkd.service**(8) allow this. Note that **systemd-hostnamed.service**(8) gives higher priority to the static hostname, so the transient hostname will only be used if the static hostname is not configured.
- Otherwise, a fallback hostname configured at compilation time will be used ("localhost").

Effectively, the static hostname has higher priority than a transient hostname, which has higher priority than the fallback hostname. Transient hostnames are equivalent, so setting a new transient hostname causes the previous transient hostname to be forgotten. The hostname specified on the kernel command line is like a transient hostname, with the exception that it has higher priority when the machine boots. Also note that those are the semantics implemented by **systemd** tools, but other programs may also set the hostname.

HISTORY

The simple configuration file format of /etc/hostname originates from Debian GNU/Linux.

SEE ALSO

systemd(1), **sethostname**(2), **hostname**(1), **hostname**(7), **machine-id**(5), **machine-info**(5),
hostnamectl(1), **systemd-hostnamed.service**(8), **systemd-firstboot**(1)

NOTES

1. NetworkManager
<https://developer.gnome.org/NetworkManager/stable/>

NAME

hostname – hostname resolution description

DESCRIPTION

Hostnames are domains, where a domain is a hierarchical, dot-separated list of subdomains; for example, the machine "monet", in the "example" subdomain of the "com" domain would be represented as "monet.example.com".

Each element of the hostname must be from 1 to 63 characters long and the entire hostname, including the dots, can be at most 253 characters long. Valid characters for hostnames are **ASCII**(7) letters from *a* to *z*, the digits from *0* to *9*, and the hyphen (–). A hostname may not start with a hyphen.

Hostnames are often used with network client and server programs, which must generally translate the name to an address for use. (This task is generally performed by either **getaddrinfo**(3) or the obsolete **gethostbyname**(3).)

Hostnames are resolved by the NSS framework in glibc according to the **hosts** configuration in **nsswitch.conf**. The DNS-based name resolver (in the **dns** NSS service module) resolves them in the following fashion.

If the name consists of a single component, that is, contains no dot, and if the environment variable **HOSTALIASES** is set to the name of a file, that file is searched for any string matching the input hostname. The file should consist of lines made up of two white-space separated strings, the first of which is the hostname alias, and the second of which is the complete hostname to be substituted for that alias. If a case-insensitive match is found between the hostname to be resolved and the first field of a line in the file, the substituted name is looked up with no further processing.

If the input name ends with a trailing dot, the trailing dot is removed, and the remaining name is looked up with no further processing.

If the input name does not end with a trailing dot, it is looked up by searching through a list of domains until a match is found. The default search list includes first the local domain, then its parent domains with at least 2 name components (longest first). For example, in the domain cs.example.com, the name lithium.cchem will be checked first as lithium.cchem.cs.example and then as lithium.cchem.example.com. lithium.cchem.com will not be tried, as there is only one component remaining from the local domain. The search path can be changed from the default by a system-wide configuration file (see **resolver**(5)).

SEE ALSO

getaddrinfo(3), **gethostbyname**(3), **nsswitch.conf**(5), **resolver**(5), **mailaddr**(7), **named**(8)

IETF RFC 1123 <<http://www.ietf.org/rfc/rfc1123.txt>>

IETF RFC 1178 <<http://www.ietf.org/rfc/rfc1178.txt>>

NAME

hostnamectl – Control the system hostname

SYNOPSIS

hostnamectl [OPTIONS...] {COMMAND}

DESCRIPTION

hostnamectl may be used to query and change the system hostname and related settings.

systemd-hostnamed.service(8) and this tool distinguish three different hostnames: the high-level "pretty" hostname which might include all kinds of special characters (e.g. "Lennart's Laptop"), the "static" hostname which is the user-configured hostname (e.g. "lennarts-laptop"), and the transient hostname which is a fallback value received from network configuration (e.g. "node12345678"). If a static hostname is set to a valid value, then the transient hostname is not used.

Note that the pretty hostname has little restrictions on the characters and length used, while the static and transient hostnames are limited to the usually accepted characters of Internet domain names, and 64 characters at maximum (the latter being a Linux limitation).

Use **systemd-firstboot**(1) to initialize the system hostname for mounted (but not booted) system images.

COMMANDS

The following commands are understood:

status

Show system hostname and related information. If no command is specified, this is the implied default.

hostname [*NAME*]

If no argument is given, print the system hostname. If an optional argument *NAME* is provided then the command changes the system hostname to *NAME*. By default, this will alter the pretty, the static, and the transient hostname alike; however, if one or more of **--static**, **--transient**, **--pretty** are used, only the selected hostnames are changed. If the pretty hostname is being set, and static or transient are being set as well, the specified hostname will be simplified in regards to the character set used before the latter are updated. This is done by removing special characters and spaces. This ensures that the pretty and the static hostname are always closely related while still following the validity rules of the specific name. This simplification of the hostname string is not done if only the transient and/or static hostnames are set, and the pretty hostname is left untouched.

The static and transient hostnames must each be either a single DNS label (a string composed of 7-bit ASCII lower-case characters and no spaces or dots, limited to the format allowed for DNS domain name labels), or a sequence of such labels separated by single dots that forms a valid DNS FQDN. The hostname must be at most 64 characters, which is a Linux limitation (DNS allows longer names).

icon-name [*NAME*]

If no argument is given, print the icon name of the system. If an optional argument *NAME* is provided then the command changes the icon name to *NAME*. The icon name is used by some graphical applications to visualize this host. The icon name should follow the [Icon Naming Specification](#)^[1].

chassis [*TYPE*]

If no argument is given, print the chassis type. If an optional argument *TYPE* is provided then the command changes the chassis type to *TYPE*. The chassis type is used by some graphical applications to visualize the host or alter user interaction. Currently, the following chassis types are defined: "desktop", "laptop", "convertible", "server", "tablet", "handset", "watch", "embedded", as well as the special chassis types "vm" and "container" for virtualized systems that lack an immediate physical chassis.

deployment [*ENVIRONMENT*]

If no argument is given, print the deployment environment. If an optional argument *ENVIRONMENT* is provided then the command changes the deployment environment to *ENVIRONMENT*. Argument *ENVIRONMENT* must be a single word without any control characters. One of the following is

suggested: "development", "integration", "staging", "production".

location [LOCATION]

If no argument is given, print the location string for the system. If an optional argument *LOCATION* is provided then the command changes the location string for the system to *LOCATION*. Argument *LOCATION* should be a human-friendly, free-form string describing the physical location of the system, if it is known and applicable. This may be as generic as "Berlin, Germany" or as specific as "Left Rack, 2nd Shelf".

OPTIONS

The following options are understood:

--no-ask-password

Do not query the user for authentication for privileged operations.

--static, --transient, --pretty

If **status** is invoked (or no explicit command is given) and one of these switches is specified, **hostnamectl** will print out just this selected hostname.

If used with **set-hostname**, only the selected hostname(s) will be updated. When more than one of these switches are specified, all the specified hostnames will be updated.

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":" , and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--json=MODE

Shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks), "pretty" (for a pretty version of the same, with indentation and line breaks) or "off" (to turn off JSON output, the default).

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

SEE ALSO

systemd(1), hostname(1), hostname(5), machine-info(5), systemctl(1), systemd-hostnamed.service(8), systemd-firstboot(1)

NOTES

- Icon Naming Specification

<http://standards.freedesktop.org/icon-naming-spec/icon-naming-spec-latest.html>

NAME

hpfsck — check integrity of an HFS+ volume

SYNOPSIS

hpfsck [**-v**] *source-path*

Description

hpfsck checks the integrity of an HFS+ volume. With the **-v** option, the data contained in the volume header is output as well.

See also

hfsplus(7), hpmount(1), hpls(1), hpcd(1), hprm(1), hpmkdir(1), hppwd(1), hpcopy(1), hpumount(1).

Author

This manual page was written by Jens Schmalzing <jensen@debian.org> for **Debian GNU/Linux** using the manual page by Klaus Halfmann <halfmann@libra.de> that comes with the source code and documentation from the **Tech Info Library**.

NAME

id – print real and effective user and group IDs

SYNOPSIS

id [*OPTION*]... [*USER*]...

DESCRIPTION

Print user and group information for each specified *USER*, or (when *USER* omitted) for the current user.

-a ignore, for compatibility with other versions

-Z, --context print only the security context of the process

-g, --group print only the effective group ID

-G, --groups print all group IDs

-n, --name print a name instead of a number, for **-ugG**

-r, --real print the real ID instead of the effective ID, with **-ugG**

-u, --user print only the effective user ID

-z, --zero delimit entries with NUL characters, not whitespace;
not permitted in default format

--help display this help and exit

--version output version information and exit

Without any OPTION, print some useful set of identified information.

AUTHOR

Written by Arnold Robbins and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/id>>
or available locally via: info '(coreutils) id invocation'

NAME

`system_data_types` – overview of system data types

DESCRIPTION

aiocb

Include: `<aio.h>`.

```
struct aiocb {
    int             aio_fildes;      /* File descriptor */
    off_t           aio_offset;     /* File offset */
    volatile void * aio_buf;        /* Location of buffer */
    size_t          aio_nbytes;     /* Length of transfer */
    int             aio_reqprio;    /* Request priority offset */
    struct sigevent aio_sigevent;  /* Signal number and value */
    int             aio_lio_opcode; /* Operation to be performed */
};
```

For further information about this structure, see [aio\(7\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [aio_cancel\(3\)](#), [aio_error\(3\)](#), [aio_fsync\(3\)](#), [aio_read\(3\)](#), [aio_return\(3\)](#), [aio_suspend\(3\)](#), [aio_write\(3\)](#), [lio_listio\(3\)](#)

clock_t

Include: `<time.h>` or `<sys/types.h>`. Alternatively, `<sys/time.h>`.

Used for system time in clock ticks or **CLOCKS_PER_SEC** (defined in `<time.h>`). According to POSIX, it shall be an integer type or a real-floating type.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [times\(2\)](#), [clock\(3\)](#)

clockid_t

Include: `<sys/types.h>`. Alternatively, `<time.h>`.

Used for clock ID type in the clock and timer functions. According to POSIX, it shall be defined as an arithmetic type.

Conforming to: POSIX.1-2001 and later.

See also: [clock_adjtime\(2\)](#), [clock_getres\(2\)](#), [clock_nanosleep\(2\)](#), [timer_create\(2\)](#), [clock_getcpuclockid\(3\)](#)

dev_t

Include: `<sys/types.h>`. Alternatively, `<sys/stat.h>`.

Used for device IDs. According to POSIX, it shall be an integer type. For further details of this type, see [makedev\(3\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [mknod\(2\)](#), [stat\(2\)](#)

div_t

Include: `<stdlib.h>`.

```
typedef struct {
    int quot; /* Quotient */
    int rem; /* Remainder */
} div_t;
```

It is the type of the value returned by the [div\(3\)](#) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [div\(3\)](#)

double_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *double*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *double_t* is *double*.
- 1 *double_t* is *double*.
- 2 *double_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *double_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *float_t* type in this page.

fd_set

Include: <sys/select.h>. Alternatively, <sys/time.h>.

A structure type that can represent a set of file descriptors. According to POSIX, the maximum number of file descriptors in an *fd_set* structure is the value of the macro **FD_SETSIZE**.

Conforming to: POSIX.1-2001 and later.

See also: **select**(2)

fenv_t

Include: <fenv.h>.

This type represents the entire floating-point environment, including control modes and status flags; for further details, see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

fexcept_t

Include: <fenv.h>.

This type represents the floating-point status flags collectively; for further details see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

FILE

Include: <stdio.h>. Alternatively, <wchar.h>.

An object type used for streams.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fclose**(3), **flockfile**(3), **fopen**(3), **fprintf**(3), **fread**(3), **fscanf**(3), **stdin**(3), **stdio**(3)

float_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *float*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *float_t* is *float*.
- 1 *float_t* is *double*.
- 2 *float_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *float_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *double_t* type in this page.

gid_t

Include: <sys/types.h>. Alternatively, <grp.h>, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold group IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: chown(2), getgid(2), getegid(2), getgroups(2), getresgid(2), getgrnam(2), credentials(7)

id_t

Include: <sys/types.h>. Alternatively, <sys/resource.h>.

A type used to hold a general identifier. According to POSIX, this shall be an integer type that can be used to contain a *pid_t*, *uid_t*, or *gid_t*.

Conforming to: POSIX.1-2001 and later.

See also: getpriority(2), waitid(2)

imaxdiv_t

Include: <inttypes.h>.

```
typedef struct {
    intmax_t      quot; /* Quotient */
    intmax_t      rem;   /* Remainder */
} imaxdiv_t;
```

It is the type of the value returned by the imaxdiv(3) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: imaxdiv(3)

intmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

A signed integer type capable of representing any value of any signed integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [INTMAX_MIN, INTMAX_MAX].

The macro INTMAX_C() expands its argument to an integer constant of type *intmax_t*.

The length modifier for *intmax_t* for the printf(3) and the scanf(3) families of functions is **j**; resulting commonly in %**jd** or %**ji** for printing *intmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *intmax_t* is not large enough to represent values of type __int128 in implementations where __int128 is defined and long long is less than 128 bits wide.

See also: the *uintmax_t* type in this page.

intN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

int8_t, *int16_t*, *int32_t*, *int64_t*

A signed integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [INTN_MIN, INTN_MAX], substituting N by the appropriate number.

According to POSIX, *int8_t*, *int16_t*, and *int32_t* are required; *int64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *intN_t* types for the printf(3) family of functions are expanded by macros of the forms **PRIIdN** and **PRIiN** (defined in <inttypes.h>); resulting for example in %"**PRI64**" or %"**PRIi64**" for printing *int64_t* values. The length modifiers for the *intN_t*

types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNdN** and **SCNiN**, (defined in *<inttypes.h>*); resulting for example in **%"SCNd8"** or **%"SCNi8"** for scanning *int8_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *uintN_t*, and *uintmax_t* types in this page.

intptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

A signed integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [**INTPTR_MIN**, **INTPTR_MAX**].

The length modifier for *intptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIIdPTR** and **PRIiPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIIdPTR"** or **%"PRIiPTR"** for printing *intptr_t* values. The length modifier for *intptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNdPTR** and **SCNiPTR**, (defined in *<inttypes.h>*); resulting commonly in **%"SCNdPTR"** or **%"SCNiPTR"** for scanning *intptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *uintptr_t* and *void ** types in this page.

lconv

Include: *<locale.h>*.

```
struct lconv {                                /* Values in the "C" locale: */
    char    *decimal_point;                  /* ". " */
    char    *thousands_sep;                 /* " " */
    char    *grouping;                      /* " " */
    char    *mon_decimal_point;             /* " " */
    char    *mon_thousands_sep;             /* " " */
    char    *mon_grouping;                  /* " " */
    char    *positive_sign;                 /* " " */
    char    *negative_sign;                 /* " " */
    char    *currency_symbol;               /* " " */
    char    frac_digits;                   /* CHAR_MAX */
    char    p_cs_precedes;                 /* CHAR_MAX */
    char    n_cs_precedes;                 /* CHAR_MAX */
    char    p_sep_by_space;                /* CHAR_MAX */
    char    n_sep_by_space;                /* CHAR_MAX */
    char    p_sign_posn;                   /* CHAR_MAX */
    char    n_sign_posn;                   /* CHAR_MAX */
    char    *int_curr_symbol;              /* " " */
    char    int_frac_digits;               /* CHAR_MAX */
    char    int_p_cs_precedes;             /* CHAR_MAX */
    char    int_n_cs_precedes;             /* CHAR_MAX */
    char    int_p_sep_by_space;             /* CHAR_MAX */
    char    int_n_sep_by_space;             /* CHAR_MAX */
    char    int_p_sign_posn;                /* CHAR_MAX */
    char    int_n_sign_posn;                /* CHAR_MAX */
};
```

Contains members related to the formatting of numeric values. In the "C" locale, its members have the values shown in the comments above.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **setlocale**(3), **localeconv**(3), **charsets**(5), **locale**(7)

ldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long      quot; /* Quotient */
    long      rem;   /* Remainder */
} ldiv_t;
```

It is the type of the value returned by the **ldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **ldiv(3)**

lldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long long  quot; /* Quotient */
    long long  rem;  /* Remainder */
} lldiv_t;
```

It is the type of the value returned by the **lldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **lldiv(3)**

off_t

Include: <sys/types.h>. Alternatively, <aio.h>, <fcntl.h>, <stdio.h>, <sys/mman.h>, <sys/stat.h>, or <unistd.h>.

Used for file sizes. According to POSIX, this shall be a signed integer type.

Versions: <aio.h> and <stdio.h> define *off_t* since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

Notes: On some architectures, the width of this type can be controlled with the feature test macro **_FILE_OFFSET_BITS**.

See also: **lseek(2)**, **mmap(2)**, **posix_fadvise(2)**, **pread(2)**, **truncate(2)**, **fseeko(3)**, **lockf(3)**, **posix_fallocate(3)**, **feature_test_macros(7)**

pid_t

Include: <sys/types.h>. Alternatively, <fcntl.h>, <sched.h>, <signal.h>, <spawn.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/wait.h>, <termios.h>, <time.h>, <unistd.h>, or <utmpx.h>.

This type is used for storing process IDs, process group IDs, and session IDs. According to POSIX, it shall be a signed integer type, and the implementation shall support one or more programming environments where the width of *pid_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: **fork(2)**, **getpid(2)**, **getppid(2)**, **getsid(2)**, **gettid(2)**, **getpgid(2)**, **kill(2)**, **pidfd_open(2)**, **sched_setscheduler(2)**, **waitpid(2)**, **sigqueue(3)**, **credentials(7)**,

ptrdiff_t

Include: <stddef.h>.

Used for a count of elements, and array indices. It is the result of subtracting two pointers. According to the C language standard, it shall be a signed integer type capable of storing values in the range [**PTRDIFF_MIN**, **PTRDIFF_MAX**].

The length modifier for *ptrdiff_t* for the **printf(3)** and the **scanf(3)** families of functions is **t**; resulting commonly in **%td** or **%ti** for printing *ptrdiff_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the `size_t` and `ssize_t` types in this page.

regex_t

Include: <`regex.h`>.

```
typedef struct {
    size_t re_nsub; /* Number of parenthesized subexpressions. */
} regex_t;
```

This is a structure type used in regular expression matching. It holds a compiled regular expression, compiled with `regcomp(3)`.

Conforming to: POSIX.1-2001 and later.

See also: `regex(3)`

regmatch_t

Include: <`regex.h`>.

```
typedef struct {
    regoff_t rm_so; /* Byte offset from start of string
                      to start of substring */
    regoff_t rm_eo; /* Byte offset from start of string of
                      the first character after the end of
                      substring */
} regmatch_t;
```

This is a structure type used in regular expression matching.

Conforming to: POSIX.1-2001 and later.

See also: `regexec(3)`

regoff_t

Include: <`regex.h`>.

According to POSIX, it shall be a signed integer type capable of storing the largest value that can be stored in either a `ptrdiff_t` type or a `ssize_t` type.

Versions: Prior to POSIX.1-2008, the type was capable of storing the largest value that can be stored in either an `off_t` type or a `ssize_t` type.

Conforming to: POSIX.1-2001 and later.

See also: the `regmatch_t` structure and the `ptrdiff_t` and `ssize_t` types in this page.

sigevent

Include: <`signal.h`>. Alternatively, <`aio.h`>, <`mqueue.h`>, or <`time.h`>.

```
struct sigevent {
    int           sigev_notify; /* Notification type */
    int           sigev_signo;  /* Signal number */
    union sigval  sigev_value; /* Signal value */
    void         (*sigev_notify_function)(union sigval);
                  /* Notification function */
    pthread_attr_t *sigev_notify_attributes;
                  /* Notification attributes */
};
```

For further details about this type, see `sigevent(7)`.

Versions: <`aio.h`> and <`time.h`> define `sigevent` since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

See also: `timer_create(2)`, `getaddrinfo_a(3)`, `lio_listio(3)`, `mq_notify(3)`

See also the *aiocb* structure in this page.

siginfo_t

Include: <signal.h>. Alternatively, <sys/wait.h>.

```
typedef struct {
    int      si_signo; /* Signal number */
    int      si_code;  /* Signal code */
    pid_t   si_pid;   /* Sending process ID */
    uid_t   si_uid;   /* Real user ID of sending process */
    void    *si_addr; /* Address of faulting instruction */
    int     si_status; /* Exit value or signal */
    union sigval si_value; /* Signal value */
} siginfo_t;
```

Information associated with a signal. For further details on this structure (including additional, Linux-specific fields), see **sigaction**(2).

Conforming to: POSIX.1-2001 and later.

See also: **pidfd_send_signal**(2), **rt_sigqueueinfo**(2), **sigaction**(2), **sigwaitinfo**(2), **psiginfo**(3)

sigset_t

Include: <signal.h>. Alternatively, <spawn.h>, or <sys/select.h>.

This is a type that represents a set of signals. According to POSIX, this shall be an integer or structure type.

Conforming to: POSIX.1-2001 and later.

See also: **epoll_pwait**(2), **ppoll**(2), **pselect**(2), **sigaction**(2), **signalfd**(2), **sigpending**(2), **sigprocmask**(2), **sigsuspend**(2), **sigwaitinfo**(2), **signal**(7)

sigval

Include: <signal.h>.

```
union sigval {
    int     sigval_int; /* Integer value */
    void   *sigval_ptr; /* Pointer value */
};
```

Data passed with a signal.

Conforming to: POSIX.1-2001 and later.

See also: **pthread_sigqueue**(3), **sigqueue**(3), **sigevent**(7)

See also the *sigevent* structure and the *siginfo_t* type in this page.

size_t

Include: <stddef.h> or <sys/types.h>. Alternatively, <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <monetary.h>, <mqueue.h>, <ndbm.h>, <pwd.h>, <regex.h>, <search.h>, <signal.h>, <stdio.h>, <stdlib.h>, <string.h>, <strings.h>, <sys/mman.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/socket.h>, <sys/uio.h>, <time.h>, <unistd.h>, <wchar.h>, or <wctype.h>.

Used for a count of bytes. It is the result of the *sizeof* operator. According to the C language standard, it shall be an unsigned integer type capable of storing values in the range [0, **SIZE_MAX**]. According to POSIX, the implementation shall support one or more programming environments where the width of *size_t* is no greater than the width of the type *long*.

The length modifier for *size_t* for the **printf**(3) and the **scanf**(3) families of functions is **z**; resulting commonly in **%zu** or **%zx** for printing *size_t* values.

Versions: <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <mqueue.h>, <pwd.h>, <signal.h>, and <sys/socket.h> define *size_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **read(2)**, **write(2)**, **fread(3)**, **fwrite(3)**, **memcmp(3)**, **memcpy(3)**, **memset(3)**, **off-setof(3)**

See also the *ptrdiff_t* and *ssize_t* types in this page.

ssize_t

Include: <sys/types.h>. Alternatively, <aio.h>, <monetary.h>, <mqueue.h>, <stdio.h>, <sys/msg.h>, <sys/socket.h>, <sys/uio.h>, or <unistd.h>.

Used for a count of bytes or an error indication. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, **SSIZE_MAX**], and the implementation shall support one or more programming environments where the width of *ssize_t* is no greater than the width of the type *long*.

Glibc and most other implementations provide a length modifier for *ssize_t* for the **printf(3)** and the **scanf(3)** families of functions, which is **z**; resulting commonly in **%zd** or **%zi** for printing *ssize_t* values. Although **z** works for *ssize_t* on most implementations, portable POSIX programs should avoid using it—for example, by converting the value to *intmax_t* and using its length modifier (**j**).

Conforming to: POSIX.1-2001 and later.

See also: **read(2)**, **readlink(2)**, **readv(2)**, **recv(2)**, **send(2)**, **write(2)**

See also the *ptrdiff_t* and *size_t* types in this page.

suseconds_t

Include: <sys/types.h>. Alternatively, <sys/select.h>, or <sys/time.h>.

Used for time in microseconds. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, 1000000], and the implementation shall support one or more programming environments where the width of *suseconds_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: the *timeval* structure in this page.

time_t

Include: <time.h> or <sys/types.h>. Alternatively, <sched.h>, <sys/msg.h>, <sys/select.h>, <sys/sem.h>, <sys/shm.h>, <sys/stat.h>, <sys/time.h>, or <utime.h>.

Used for time in seconds. According to POSIX, it shall be an integer type.

Versions: <sched.h> defines *time_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **stime(2)**, **time(2)**, **ctime(3)**, **difftime(3)**

timer_t

Include: <sys/types.h>. Alternatively, <time.h>.

Used for timer ID returned by **timer_create(2)**. According to POSIX, there are no defined comparison or assignment operators for this type.

Conforming to: POSIX.1-2001 and later.

See also: **timer_create(2)**, **timer_delete(2)**, **timer_getoverrun(2)**, **timer_settime(2)**

timespec

Include: <time.h>. Alternatively, <aio.h>, <mqueue.h>, <sched.h>, <signal.h>, <sys/select.h>, or <sys/stat.h>.

```
struct timespec {
    time_t tv_sec; /* Seconds */
```

```
    long      tv_nsec; /* Nanoseconds */
}
```

Describes times in seconds and nanoseconds.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **clock_gettime(2)**, **clock_nanosleep(2)**, **nanosleep(2)**, **timerfd_gettime(2)**, **timer_gettime(2)**

timeval

Include: <sys/time.h>. Alternatively, <sys/resource.h>, <sys/select.h>, or <utmpx.h>.

```
struct timeval {
    time_t      tv_sec;   /* Seconds */
    suseconds_t tv_usec; /* Microseconds */
};
```

Describes times in seconds and microseconds.

Conforming to: POSIX.1-2001 and later.

See also: **gettimeofday(2)**, **select(2)**, **utimes(2)**, **adjtime(3)**, **futimes(3)**, **timeradd(3)**

uid_t

Include: <sys/types.h>. Alternatively, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold user IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: **chown(2)**, **getuid(2)**, **geteuid(2)**, **getresuid(2)**, **getpwnam(2)**, **credentials(7)**

uintmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

An unsigned integer type capable of representing any value of any unsigned integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [0, **UINTMAX_MAX**].

The macro **UINTMAX_C()** expands its argument to an integer constant of type *uintmax_t*.

The length modifier for *uintmax_t* for the **printf(3)** and the **scanf(3)** families of functions is **j**; resulting commonly in **%ju** or **%jx** for printing *uintmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *uintmax_t* is not large enough to represent values of type *unsigned __int128* in implementations where *unsigned __int128* is defined and *unsigned long long* is less than 128 bits wide.

See also: the *intmax_t* type in this page.

uintN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

uint8_t, *uint16_t*, *uint32_t*, *uint64_t*

An unsigned integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [0, **UINTN_MAX**], substituting N by the appropriate number.

According to POSIX, *uint8_t*, *uint16_t*, and *uint32_t* are required; *uint64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *uintN_t* types for the **printf(3)** family of functions are expanded by macros of the forms **PRIuN**, **PRIoN**, **PRIxN**, and **PRIXN** (defined in <inttypes.h>); resulting for example in **%"PRIu32"** or **%"PRIx32"** for printing *uint32_t* values. The length modifiers for

the *uintN_t* types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNuN**, **SCNoN**, **SCNxN**, and **SCNXN** (defined in *<inttypes.h>*); resulting for example in **%"SCNu16"** or **%"SCNx16"** for scanning *uint16_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *intN_t*, and *uintmax_t* types in this page.

uintptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

An unsigned integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [0, **UINTPTR_MAX**].

The length modifier for *uintptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIupTR**, **PRIoPTR**, **PRIxPTR**, and **PRIxPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIupTR"** or **%"PRIxPTR"** for printing *uintptr_t* values. The length modifier for *uintptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNuPTR**, **SCNoPTR**, **SCNxPTR**, and **SCNXPTR** (defined in *<inttypes.h>*); resulting commonly in **%"SCNuPTR"** or **%"SCNxPTR"** for scanning *uintptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intptr_t* and *void ** types in this page.

va_list

Include: *<stdarg.h>*. Alternatively, *<stdio.h>*, or *<wchar.h>*.

Used by functions with a varying number of arguments of varying types. The function must declare an object of type *va_list* which is used by the macros **va_start**(3), **va_arg**(3), **va_copy**(3), and **va_end**(3) to traverse the list of arguments.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **va_start**(3), **va_arg**(3), **va_copy**(3), **va_end**(3)

*void **

According to the C language standard, a pointer to any object type may be converted to a pointer to *void* and back. POSIX further requires that any pointer, including pointers to functions, may be converted to a pointer to *void* and back.

Conversions from and to any other pointer type are done implicitly, not requiring casts at all. Note that this feature prevents any kind of type checking: the programmer should be careful not to convert a *void ** value to a type incompatible to that of the underlying data, because that would result in undefined behavior.

This type is useful in function parameters and return value to allow passing values of any type. The function will typically use some mechanism to know the real type of the data being passed via a pointer to *void*.

A value of this type can't be dereferenced, as it would give a value of type *void*, which is not possible. Likewise, pointer arithmetic is not possible with this type. However, in GNU C, pointer arithmetic is allowed as an extension to the standard; this is done by treating the size of a *void* or of a function as 1. A consequence of this is that *sizeof* is also allowed on *void* and on function types, and returns 1.

The conversion specifier for *void ** for the **printf**(3) and the **scanf**(3) families of functions is **p**.

Versions: The POSIX requirement about compatibility between *void ** and function pointers was added in POSIX.1-2008 Technical Corrigendum 1 (2013).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **malloc**(3), **memcmp**(3), **memcpy**(3), **memset**(3)

See also the *intptr_t* and *uintptr_t* types in this page.

NOTES

The structures described in this manual page shall contain, at least, the members shown in their definition, in no particular order.

Most of the integer types described in this page don't have a corresponding length modifier for the `printf(3)` and the `scanf(3)` families of functions. To print a value of an integer type that doesn't have a length modifier, it should be converted to `intmax_t` or `uintmax_t` by an explicit cast. To scan into a variable of an integer type that doesn't have a length modifier, an intermediate temporary variable of type `intmax_t` or `uintmax_t` should be used. When copying from the temporary variable to the destination variable, the value could overflow. If the type has upper and lower limits, the user should check that the value is within those limits, before actually copying the value. The example below shows how these conversions should be done.

Conventions used in this page

In "Conforming to" we only concern ourselves with C99 and later and POSIX.1-2001 and later. Some types may be specified in earlier versions of one of these standards, but in the interests of simplicity we omit details from earlier standards.

In "Include", we first note the "primary" header(s) that define the type according to either the C or POSIX.1 standards. Under "Alternatively", we note additional headers that the standards specify shall define the type.

EXAMPLES

The program shown below scans from a string and prints a value stored in a variable of an integer type that doesn't have a length modifier. The appropriate conversions from and to `intmax_t`, and the appropriate range checks, are used as explained in the notes section above.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int
main (void)
{
    static const char *const str = "500000 us in half a second";
    suseconds_t us;
    intmax_t tmp;

    /* Scan the number from the string into the temporary variable */

    sscanf(str, "%jd", &tmp);

    /* Check that the value is within the valid range of suseconds_t */

    if (tmp < -1 || tmp > 1000000) {
        fprintf(stderr, "Scanned value outside valid range!\n");
        exit(EXIT_FAILURE);
    }

    /* Copy the value to the suseconds_t variable 'us' */

    us = tmp;

    /* Even though suseconds_t can hold the value -1, this isn't
       a sensible number of microseconds */
}
```

```
if (us < 0) {
    fprintf(stderr, "Scanned value shouldn't be negative!\n");
    exit(EXIT_FAILURE);
}

/* Print the value */

printf("There are %jd microseconds in half a second.\n",
       (intmax_t) us);

exit(EXIT_SUCCESS);
}
```

SEE ALSO

[feature_test_macros\(7\)](#), [standards\(7\)](#)

COLPHON

This page is part of release 5.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.

NAME

ifcfg – simplistic script which replaces ifconfig IP management

SYNOPSIS

ifcfg [*DEVICE*] [*command*] *ADDRESS* [*PEER*]

DESCRIPTION

This manual page documents briefly the **ifcfg** command.

This is a simplistic script replacing one option of **ifconfig**, namely, IP address management. It not only adds addresses, but also carries out Duplicate Address Detection RFC-DHCP, sends unsolicited ARP to update the caches of other hosts sharing the interface, adds some control routes and restarts Router Discovery when it is necessary.

IFCONFIG - COMMAND SYNTAX**DEVICE**

- it may have alias, suffix, separated by colon.

command

- add, delete or stop.

ADDRESS

- optionally followed by prefix length.

peer - optional peer address for pointpoint interfaces.

NOTES

This script is not suitable for use with IPv6.

SEE ALSO

IP Command reference **ip-cref.ps**

NAME

ifconfig – configure a network interface

SYNOPSIS

```
ifconfig [-v] [-a] [-s] [interface]
ifconfig [-v] interface [aftype] options | address ...
```

DESCRIPTION

Ifconfig is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, **ifconfig** displays the status of the currently active interfaces. If a single **interface** argument is given, it displays the status of the given interface only; if a single **-a** argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

Address Families

If the first argument after the interface name is recognized as the name of a supported address family, that address family is used for decoding and displaying all protocol addresses. Currently supported address families include **inet** (TCP/IP, default), **inet6** (IPv6), **ax25** (AMPR Packet Radio), **ddp** (Appletalk Phase 2), **ipx** (Novell IPX) and **netrom** (AMPR Packet radio). All numbers supplied as parts in IPv4 dotted decimal notation may be decimal, octal, or hexadecimal, as specified in the ISO C standard (that is, a leading 0x or 0X implies hexadecimal; otherwise, a leading '0' implies octal; otherwise, the number is interpreted as decimal). Use of hexadecimal and octal numbers is not RFC-compliant and therefore its use is discouraged.

OPTIONS

- a** display all interfaces which are currently available, even if down
- s** display a short list (like netstat -i)
- v** be more verbose for some error conditions

interface

The name of the interface. This is usually a driver name followed by a unit number, for example **eth0** for the first Ethernet interface. If your kernel supports alias interfaces, you can specify them with syntax like **eth0:0** for the first alias of eth0. You can use them to assign more addresses. To delete an alias interface use **ifconfig eth0:0 down**. Note: for every scope (i.e. same net with address/netmask combination) all aliases are deleted, if you delete the first (primary).

up This flag causes the interface to be activated. It is implicitly specified if an address is assigned to the interface; you can suppress this behavior when using an alias interface by appending an **-** to the alias (e.g. **eth0:0-**). It is also suppressed when using the IPv4 0.0.0.0 address as the kernel will use this to implicitly delete alias interfaces.

down This flag causes the driver for this interface to be shut down.

[–]arp Enable or disable the use of the ARP protocol on this interface.

[–]promisc

Enable or disable the **promiscuous** mode of the interface. If selected, all packets on the network will be received by the interface.

[–]allmulti

Enable or disable **all-multicast** mode. If selected, all multicast packets on the network will be received by the interface.

mtu N This parameter sets the Maximum Transfer Unit (MTU) of an interface.

dstaddr addr

Set the remote IP address for a point-to-point link (such as PPP). This keyword is now obsolete; use the **pointopoint** keyword instead.

netmask addr

Set the IP network mask for this interface. This value defaults to the usual class A, B or C network mask (as derived from the interface IP address), but it can be set to any value.

add addr/prefixlen

Add an IPv6 address to an interface.

del addr/prefixlen

Remove an IPv6 address from an interface.

tunnel ::aa.bb.cc.dd

Create a new SIT (IPv6-in-IPv4) device, tunnelling to the given destination.

irq addr

Set the interrupt line used by this device. Not all devices can dynamically change their IRQ setting.

io_addr addr

Set the start address in I/O space for this device.

mem_start addr

Set the start address for shared memory used by this device. Only a few devices need this.

media type

Set the physical port or medium type to be used by the device. Not all devices can change this setting, and those that can vary in what values they support. Typical values for **type** are **10base2** (thin Ethernet), **10baseT** (twisted-pair 10Mbps Ethernet), **AUI** (external transceiver) and so on. The special medium type of **auto** can be used to tell the driver to auto-sense the media. Again, not all drivers can do this.

[–]broadcast [addr]

If the address argument is given, set the protocol broadcast address for this interface. Otherwise, set (or clear) the **IFF_BROADCAST** flag for the interface.

[–]pointopoint [addr]

This keyword enables the **point-to-point** mode of an interface, meaning that it is a direct link between two machines with nobody else listening on it.

If the address argument is also given, set the protocol address of the other side of the link, just like the obsolete **dstaddr** keyword does. Otherwise, set or clear the **IFF_POINTOPOINT** flag for the interface.

hw class address

Set the hardware address of this interface, if the device driver supports this operation. The keyword must be followed by the name of the hardware class and the printable ASCII equivalent of the hardware address. Hardware classes currently supported include **ether** (Ethernet), **ax25** (AMPR AX.25), **ARCnet** and **netrom** (AMPR NET/ROM).

multicast

Set the multicast flag on the interface. This should not normally be needed as the drivers set the flag correctly themselves.

address

The IP address to be assigned to this interface.

txqueuelen length

Set the length of the transmit queue of the device. It is useful to set this to small values for slower devices with a high latency (modem links, ISDN) to prevent fast bulk transfers from disturbing interactive traffic like telnet too much.

NOTES

Since kernel release 2.2 there are no explicit interface statistics for alias interfaces anymore. The statistics printed for the original address are shared with all alias addresses on the same device. If you want per-address statistics you should add explicit accounting rules for the address using the **iptables(8)** command.

Interrupt problems with Ethernet device drivers fail with EAGAIN (*SIOCSIFFLAGS: Resource temporarily unavailable*) it is most likely a interrupt conflict. See <http://www.scyld.com/expert/irq-conflict.html> for more information.

FILES

/proc/net/dev
/proc/net/if_inet6

BUGS

Ifconfig uses the ioctl access method to get the full address information, which limits hardware addresses to 8 bytes. Because Infiniband hardware address has 20 bytes, only the first 8 bytes are displayed correctly. Please use **ip link** command from **iproute2** package to display link layer informations including the hardware address.

While appletalk DDP and IPX addresses will be displayed they cannot be altered by this command.

SEE ALSO

route(8), **netstat(8)**, **arp(8)**, **rarp(8)**, **iptables(8)**, **ifup(8)**, **interfaces(5)**.
<http://physics.nist.gov/cuu/Units/binary.html> - Prefixes for binary multiples

AUTHORS

Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
Alan Cox, <Alan.Cox@linux.org>
Phil Blundell, <Philip.Blundell@pobox.com>
Andi Kleen
Bernd Eckenfels, <net-tools@lina.inka.de>

NAME

`info` – read Info documents

SYNOPSIS

`info [OPTION]... [MENU-ITEM ...]`

DESCRIPTION

Read documentation in Info format.

Frequently-used options:

-a, --all

use all matching manuals

-k, --apropos=STRING

look up STRING in all indices of all manuals

-d, --directory=DIR

add DIR to INFOPATH

-f, --file=MANUAL

specify Info manual to visit

-h, --help

display this help and exit

--index-search=STRING

go to node pointed by index entry STRING

-n, --node=NODENAME

specify nodes in first visited Info file

-o, --output=FILE

output selected nodes to FILE

-O, --show-options, --usage

go to command-line options node

--subnodes

recursively output menu items

-v, --variable VAR=VALUE

assign VALUE to Info variable VAR

--version

display version information and exit

-w, --where, --location

print physical location of Info file

The first non-option argument, if present, is the menu entry to start from; it is searched for in all 'dir' files along INFOPATH. If it is not present, info merges all 'dir' files and shows the result. Any remaining arguments are treated as the names of menu items relative to the initial node visited.

For a summary of key bindings, type H within Info.

EXAMPLES

`info` show top-level dir menu

`info info-stnd`

show the manual for this Info program

`info emacs`

start at emacs node from top-level dir

`info emacs buffers`

select buffers menu entry in emacs manual

```
info emacs -n Files
      start at Files node within emacs manual
info '(emacs)Files'
      alternative way to start at Files node
info --show-options emacs
      start at node with emacs' command line options
info --subnodes -o out.txt emacs
      dump entire emacs manual to out.txt
info -f ./foo.info
      show file ./foo.info, not searching dir
```

REPORTING BUGS

Email bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org.
Texinfo home page: <http://www.gnu.org/software/texinfo/>

COPYRIGHT

Copyright © 2021 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<http://gnu.org/licenses/gpl.html>>
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

The full documentation for **info** is maintained as a Texinfo manual. If the **info** program is properly installed at your site, the command

info info

should give you access to the complete manual. (Or, if you have Emacs, M-x info will lead to the manual.)

NAME

asn1parse, ca, ciphers, cms, crl, crl2pkcs7, dgst, dhparam, dsa, dsaparam, ec, ecpam, enc, engine, errstr, gendsa, genkey, genrsa, info, kdf, mac, nseq, ocsp, passwd, pkcs12, pkcs7, pkcs8, pkey, pkeyparam, pkeyutl, prime, rand, rehash, req, rsa, rsautl, s_client, s_server, s_time, sess_id, smime, speed, spkac, srp, storeutl, ts, verify, version, x509 – OpenSSL application commands

SYNOPSIS

openssl cmd -help | [-option | -option arg] ... [arg] ...

DESCRIPTION

Every *cmd* listed above is a (sub-)command of the **openssl**(1) application. It has its own detailed manual page at **openssl-cmd**(1). For example, to view the manual page for the **openssl dgst** command, type `man openssl-dgst`.

OPTIONS

Among others, every subcommand has a help option.

-help

Print out a usage message for the subcommand.

SEE ALSO

openssl(1), **openssl-asn1parse**(1), **openssl-ca**(1), **openssl-ciphers**(1), **openssl-cms**(1), **openssl-crl**(1), **openssl-crl2pkcs7**(1), **openssl-dgst**(1), **openssl-dhparam**(1), **openssl-dsa**(1), **openssl-dsaparam**(1), **openssl-ec**(1), **openssl-ecparam**(1), **openssl-enc**(1), **openssl-engine**(1), **openssl-errstr**(1), **openssl-gendsa**(1), **openssl-genkey**(1), **openssl-genrsa**(1), **openssl-info**(1), **openssl-kdf**(1), **openssl-mac**(1), **openssl-nseq**(1), **openssl-ocsp**(1), **openssl-passwd**(1), **openssl-pkcs12**(1), **openssl-pkcs7**(1), **openssl-pkcs8**(1), **openssl-pkey**(1), **openssl-pkeyparam**(1), **openssl-pkeyutl**(1), **openssl-prime**(1), **openssl-rand**(1), **openssl-rehash**(1), **openssl-req**(1), **openssl-rsa**(1), **openssl-rsautl**(1), **openssl-s_client**(1), **openssl-s_server**(1), **openssl-s_time**(1), **openssl-sess_id**(1), **openssl-smime**(1), **openssl-speed**(1), **openssl-spkac**(1), **openssl-srp**(1), **openssl-storeutl**(1), **openssl-ts**(1), **openssl-verify**(1), **openssl-version**(1), **openssl-x509**(1),

HISTORY

Initially, the manual page entry for the **openssl cmd** command used to be available at *cmd*(1). Later, the alias **openssl-cmd**(1) was introduced, which made it easier to group the **openssl** commands using the **apropos**(1) command or the shell's tab completion.

In order to reduce cluttering of the global manual page namespace, the manual page entries without the 'openssl-' prefix have been deprecated in OpenSSL 3.0 and will be removed in OpenSSL 4.0.

COPYRIGHT

Copyright 2019–2020 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the “License”). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <<https://www.openssl.org/source/license.html>>.

NAME

info – readable online documentation

DESCRIPTION

The Info file format is an easily-parsable representation for online documents. It can be read by *emacs(1)* and *info(1)* among other programs.

Info files are usually created from *texinfo(5)* sources by *makeinfo(1)*, but can be created from scratch if so desired.

For a full description of the Texinfo language and associated tools, please see the Texinfo manual (written in Texinfo itself). Most likely, running this command from your shell:

```
info texinfo  
or this key sequence from inside Emacs:  
M-x info RET m texinfo RET  
will get you there.
```

AVAILABILITY

<http://www.gnu.org/software/texinfo/>

REPORTING BUGS

Please send bug reports to bug-texinfo@gnu.org, general questions and discussion to help-texinfo@gnu.org.

SEE ALSO

info(1), *install-info(1)*, *makeinfo(1)*, *texi2dvi(1)*,
texindex(1).
emacs(1), *tex(1)*.
texinfo(5).

NAME

insmod – Simple program to insert a module into the Linux Kernel

SYNOPSIS

insmod [*filename*] [*module options...*]

DESCRIPTION

insmod is a trivial program to insert a module into the kernel. Most users will want to use **modprobe(8)** instead, which is more clever and can handle module dependencies.

Only the most general of error messages are reported: as the work of trying to link the module is now done inside the kernel, the **dmesg** usually gives more information about errors.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

modprobe(8), **rmmmod(8)**, **lsmod(8)**, **modinfo(8)** **depmod(8)**

AUTHORS

Jon Masters <jcm@jonmasters.org>
Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>
Developer

NAME

ip-route – routing table management

SYNOPSIS

```
ip [ ip-OPTIONS ] route { COMMAND | help }
```

```
ip route { show | flush } SELECTOR
```

```
ip route save SELECTOR
```

```
ip route restore
```

```
ip route get ROUTE_GET_FLAGS ADDRESS[ fr om ADDRESS iif STRING ] [ oif STRING ] [ mark MARK ] [ tos TOS ] [ vrf NAME ] [ ippproto PROTOCOL ] [ sport NUMBER ] [ dport NUMBER ]
```

```
ip route { add | del | change | append | replace } ROUTE
```

```
SELECTOR := [ root PREFIX ] [ match PREFIX ] [ exact PREFIX ] [ table TABLE_ID ] [ vrf NAME ] [ proto RTPROTO ] [ type TYPE ] [ scope SCOPE ]
```

```
ROUTE := NODE_SPEC [ INFO_SPEC ]
```

```
NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ] [ table TABLE_ID ] [ proto RTPROTO ] [ scope SCOPE ] [ metric METRIC ] [ ttl-propagate { enabled | disabled } ]
```

```
INFO_SPEC := { NH | nhid ID } OPTIONS FLAGS [ nexthop NH ] ...
```

```
NH := [ encaps ENCAP ] [ via [ FAMILY ] ADDRESS ] [ dev STRING ] [ weight NUMBER ] NHFLAGS
```

```
FAMILY := [ inet | inet6 | mpls | bridge | link ]
```

```
OPTIONS := FLAGS [ mtu NUMBER ] [ advmss NUMBER ] [ as [ to ] ADDRESS ] rtt TIME [ rttvar TIME ] [ reordering NUMBER ] [ window NUMBER ] [ cwnd NUMBER ] [ ssthresh NUMBER ] [ realms REALM ] [ rto_min TIME ] [ initcwnd NUMBER ] [ initrwnd NUMBER ] [ features FEATURES ] [ quickack BOOL ] [ congett NAME ] [ pref PREF ] [ expires TIME ] [ fastopen_no_cookie BOOL ]
```

```
TYPE := [ unicast | local | broadcast | multicast | throw | unreachable | prohibit | blackhole | nat ]
```

```
TABLE_ID := [ local | main | default | all | NUMBER ]
```

```
SCOPE := [ host | link | global | NUMBER ]
```

```
NHFLAGS := [ onlink | pervasive ]
```

```
RTPROTO := [ kernel | boot | static | NUMBER ]
```

```
FEATURES := [ ecn | ]
```

```
PREF := [ low | medium | high ]
```

```
ENCAP := [ ENCAP_MPLS | ENCAP_IP | ENCAP_BPF | ENCAP_SEG6 | ENCAP_SEG6LOCAL | ENCAP_IOAM6 ]
```

```

ENCAP_MPLS := mpls [ LABEL ] [ ttl TTL ]
ENCAP_IP := ip id TUNNEL_ID dst REMOTE_IP [ src SRC ] [ tos TOS ] [ ttl TTL ]
ENCAP_BPF := bpf [ in PROG ] [ out PROG ] [ xmit PROG ] [ headroom SIZE ]
ENCAP_SEG6 := seg6 mode [ encaps | inline | l2encap ] segs SEGMENTS [ hmac KEYID ]
ENCAP_SEG6LOCAL := seg6local action SEG6_ACTION [ SEG6_ACTION_PARAM ] [ count ]
ENCAP_IOAM6 := ioam6 trace prealloc type IOAM6_TRACE_TYPE ns IOAM6_NAMESPACE size
IOAM6_TRACE_SIZE
ROUTE_GET_FLAGS := [ fibmatch ]
```

DESCRIPTION

ip route is used to manipulate entries in the kernel routing tables.

Route types:

unicast - the route entry describes real paths to the destinations covered by the route prefix.

unreachable - these destinations are unreachable. Packets are discarded and the ICMP message *host unreachable* is generated. The local senders get an *EHOSTUNREACH* error.

blackhole - these destinations are unreachable. Packets are discarded silently. The local senders get an *EINVAL* error.

prohibit - these destinations are unreachable. Packets are discarded and the ICMP message *communication administratively prohibited* is generated. The local senders get an *EACCES* error.

local - the destinations are assigned to this host. The packets are looped back and delivered locally.

broadcast - the destinations are broadcast addresses. The packets are sent as link broadcasts.

throw - a special control route used together with policy rules. If such a route is selected, lookup in this table is terminated pretending that no route was found. Without policy routing it is equivalent to the absence of the route in the routing table. The packets are dropped and the ICMP message *net unreachable* is generated. The local senders get an *ENETUNREACH* error.

nat - a special NAT route. Destinations covered by the prefix are considered to be

dummy (or external) addresses which require translation to real (or internal) ones before forwarding. The addresses to translate to are selected with the attribute **via**. **Warning:** Route NAT is no longer supported in Linux 2.6.

anycast - *not implemented* the destinations are *anycast* addresses assigned to this host. They are mainly equivalent to **local** with one difference: such addresses are invalid when used as the source address of any packet.

multicast - a special type used for multicast routing. It is not present in normal routing tables.

Route tables: Linux-2.x can pack routes into several routing tables identified by a number in the range from 1 to $2^{32}-1$ or by name from the file **/etc/iproute2/rt_tables** By default all normal routes are inserted into the **main** table (ID 254) and the kernel only uses this table when calculating routes. Values (0, 253, 254, and 255) are reserved for built-in use.

Actually, one other table always exists, which is invisible but even more important. It is the **local** table (ID 255). This table consists of routes for local and broadcast addresses. The kernel maintains this table automatically and the administrator usually need not modify it or even look at it.

The multiple routing tables enter the game when *policy routing* is used.

```
ip route add
    add new route
ip route change
    change route
ip route replace
    change or add new one
```

to TYPE PREFIX (default)

the destination prefix of the route. If **TYPE** is omitted, **ip** assumes type **unicast**. Other values of **TYPE** are listed above. **PREFIX** is an IP or IPv6 address optionally followed by a slash and the prefix length. If the length of the prefix is missing, **ip** assumes a full-length host route. There is also a special **PREFIX default** - which is equivalent to IP **0/0** or to IPv6 **::/0**.

tos TOS

dsfield TOS

the Type Of Service (TOS) key. This key has no associated mask and the longest match is understood as: First, compare the TOS of the route and of the packet. If they are not equal, then the packet may still match a route with a zero TOS. **TOS** is either an 8 bit hexadecimal number or an identifier from **/etc/iproute2/rt_dsfield**.

metric NUMBER

preference NUMBER

the preference value of the route. **NUMBER** is an arbitrary 32bit number, where routes with lower values are preferred.

table *TABLEID*

the table to add this route to. *TABLEID* may be a number or a string from the file */etc/iproute2/rt_tables*. If this parameter is omitted, ip assumes the **main** table, with the exception of **local**, **broadcast** and **nat** routes, which are put into the **local** table by default.

vrf *NAME*

the vrf name to add this route to. Implicitly means the table associated with the VRF.

dev *NAME*

the output device name.

via [*FAMILY*] *ADDRESS*

the address of the nexthop router, in the address family *FAMILY*. Actually, the sense of this field depends on the route type. For normal **unicast** routes it is either the true next hop router or, if it is a direct route installed in BSD compatibility mode, it can be a local address of the interface. For NAT routes it is the first address of the block of translated IP destinations.

src *ADDRESS*

the source address to prefer when sending to the destinations covered by the route prefix.

realm *REALMID*

the realm to which this route is assigned. *REALMID* may be a number or a string from the file */etc/iproute2/rt_realms*.

mtu *MTU***mtu lock** *MTU*

the MTU along the path to the destination. If the modifier **lock** is not used, the MTU may be updated by the kernel due to Path MTU Discovery. If the modifier **lock** is used, no path MTU discovery will be tried, all packets will be sent without the DF bit in IPv4 case or fragmented to MTU for IPv6.

window *NUMBER*

the maximal window for TCP to advertise to these destinations, measured in bytes. It limits maximal data bursts that our TCP peers are allowed to send to us.

rtt *TIME*

the initial RTT ('Round Trip Time') estimate. If no suffix is specified the units are raw values passed directly to the routing code to maintain compatibility with previous releases. Otherwise if a suffix of s, sec or secs is used to specify seconds and ms, msec or msecs to specify milliseconds.

rttvar *TIME* (**Linux 2.3.15+** only)

the initial RTT variance estimate. Values are specified as with **rtt** above.

rto_min *TIME* (**Linux 2.6.23+** only)

the minimum TCP Retransmission TimeOut to use when communicating with this destination. Values are specified as with **rtt** above.

ssthresh NUMBER (Linux 2.3.15+ only)

an estimate for the initial slow start threshold.

cwnd NUMBER (Linux 2.3.15+ only)

the clamp for congestion window. It is ignored if the **lock** flag is not used.

initcwnd NUMBER (Linux 2.5.70+ only)

the initial congestion window size for connections to this destination. Actual window size is this value multiplied by the MSS (“Maximal Segment Size”) for same connection. The default is zero, meaning to use the values specified in RFC2414.

initrwnd NUMBER (Linux 2.6.33+ only)

the initial receive window size for connections to this destination. Actual window size is this value multiplied by the MSS of the connection. The default value is zero, meaning to use Slow Start value.

features FEATURES (Linux 3.18+only)

Enable or disable per-route features. Only available feature at this time is **ecn** to enable explicit congestion notification when initiating connections to the given destination network. When responding to a connection request from the given network, **ecn** will also be used even if the **net.ipv4.tcp_ecn** sysctl is set to 0.

quickack BOOL (Linux 3.11+ only)

Enable or disable quick ack for connections to this destination.

fastopen_no_cookie BOOL (Linux 4.15+ only)

Enable TCP Fastopen without a cookie for connections to this destination.

congctl NAME (Linux 3.20+ only)**congctl lock NAME (Linux 3.20+ only)**

Sets a specific TCP congestion control algorithm only for a given destination. If not specified, Linux keeps the current global default TCP congestion control algorithm, or the one set from the application. If the modifier **lock** is not used, an application may nevertheless overwrite the suggested congestion control algorithm for that destination. If the modifier **lock** is used, then an application is not allowed to overwrite the specified congestion control algorithm for that destination, thus it will be enforced/guaranteed to use the proposed algorithm.

advmss NUMBER (Linux 2.3.15+ only)

the MSS ('Maximal Segment Size') to advertise to these destinations when establishing TCP connections. If it is not given, Linux uses a default value calculated from the first hop device MTU. (If the path to these destination is asymmetric, this guess may be wrong.)

reordering NUMBER (Linux 2.3.15+ only)

Maximal reordering on the path to this destination. If it is not given, Linux uses the value selected with sysctl variable **net/ipv4/tcp_reordering**.

nexthop *NEXTHOP*

the nexthop of a multipath route. *NEXTHOP* is a complex value with its own syntax similar to the top level argument lists:

via [*FAMILY*] *ADDRESS* - is the nexthop router.

dev *NAME* - is the output device.

weight *NUMBER* - is a weight for this element of a multipath route reflecting its relative bandwidth or quality.

The internal buffer used in iproute2 limits the maximum number of nexthops that may be specified in one go. If only *ADDRESS* is given, the current buffer size allows for 144 IPv6 nexthops and 253 IPv4 ones. For IPv4, this effectively limits the number of nexthops possible per route. With IPv6, further nexthops may be appended to the same route via **ip route append** command.

scope *SCOPE_VAL*

the scope of the destinations covered by the route prefix. *SCOPE_VAL* may be a number or a string from the file */etc/iproute2/rt_scopes*. If this parameter is omitted, ip assumes scope **global** for all gatewayed unicast routes, scope **link** for direct unicast and broadcast routes and scope **host** for local routes.

protocol *RTPROTO*

the routing protocol identifier of this route. *RTPROTO* may be a number or a string from the file */etc/iproute2/rt_protos*. If the routing protocol ID is not given, ip assumes protocol **boot** (i.e. it assumes the route was added by someone who doesn't understand what they are doing). Several protocol values have a fixed interpretation. Namely:

redirect - the route was installed due to an ICMP redirect.

kernel - the route was installed by the kernel during autoconfiguration.

boot - the route was installed during the bootup sequence. If a routing daemon starts, it will purge all of them.

static - the route was installed by the administrator to override dynamic routing. Routing daemon will respect them and, probably, even advertise them to its peers.

ra - the route was installed by Router Discovery protocol.

The rest of the values are not reserved and the administrator is free to assign (or not to assign) protocol tags.

onlink pretend that the nexthop is directly attached to this link, even if it does not match any interface prefix.

pref *PREF*

the IPv6 route preference. *PREF* is a string specifying the route preference as defined in RFC4191 for Router Discovery messages. Namely:

low - the route has a lowest priority

medium - the route has a default priority

high - the route has a highest priority

nhid *ID*

use nexthop object with given id as nexthop specification.

encap *ENCAPTYPE ENCAPHDR*

attach tunnel encapsulation attributes to this route.

ENCAPTYPE is a string specifying the supported encapsulation type. Namely:

mpls - encapsulation type MPLS

ip - IP encapsulation (Geneve, GRE, VXLAN, ...)

bpf - Execution of BPF program

seg6 - encapsulation type IPv6 Segment Routing

seg6local - local SRv6 segment processing

ioam6 - encapsulation type IPv6 IOAM

ENCAPHDR is a set of encapsulation attributes specific to the *ENCAPTYPE*.

mpls

MPLSLABEL - mpls label stack with labels separated by /

ttl *TTL* - TTL to use for MPLS header or 0 to inherit from IP header

ip

id *TUNNEL_ID* **dst** *REMOTE_IP* [**src** *SRC*] [**tos** *TOS*] [**ttl** *TTL*] [**key**] [**csum**] [**seq**]

bpf

in *PROG* - BPF program to execute for incoming packets

out PROG - BPF program to execute for outgoing packets

xmit PROG - BPF program to execute for transmitted packets

headroom SIZE - Size of header BPF program will attach
(xmit)

seg6

mode inline - Directly insert Segment Routing Header after
IPv6 header

mode encaps - Encapsulate packet in an outer IPv6 header with
SRH

mode l2encap - Encapsulate ingress L2 frame within an outer
IPv6 header and SRH

SEGMENTS - List of comma-separated IPv6 addresses

KEYID - Numerical value in decimal representation. See **ip-
sr(8)**.

seg6local

SEG6_ACTION [SEG6_ACTION_PARAM] [count] - Operation to perform on matching packets. The optional **count** attribute is used to collect statistics on the processing of actions. Three counters are implemented: 1) packets correctly processed; 2) bytes correctly processed; 3) packets that cause a processing error (i.e., missing SID List, wrong SID List, etc). To retrieve the counters related to an action use the **-s** flag in the **show** command. The following actions are currently supported (**Linux 4.14+ only**).

End - Regular SRv6 processing as intermediate segment endpoint. This action only accepts packets with a non-zero Segments Left value. Other matching packets are dropped.

End.X nh6 NEXTHOP - Regular SRv6 processing as intermediate segment endpoint. Additionally, forward processed packets to given next-hop. This action only accepts packets with a non-zero Segments Left value. Other matching packets are dropped.

End.DX6 nh6 NEXTHOP - Decapsulate inner IPv6 packet and forward it to the specified next-hop. If the argument is set to ::, then the next-hop is selected according to the local

selection rules. This action only accepts packets with either a zero Segments Left value or no SRH at all, and an inner IPv6 packet. Other matching packets are dropped.

End.DT6 { table | vrftable } TABLEID - Decapsulate the inner IPv6 packet and forward it according to the specified lookup table. *TABLEID* is either a number or a string from the file */etc/iproute2/rt_tables*. If *vrftable* is used, the argument must be a VRF device associated with the table id. Moreover, the VRF table associated with the table id must be configured with the VRF strict mode turned on (net.vrf.strict_mode=1). This action only accepts packets with either a zero Segments Left value or no SRH at all, and an inner IPv6 packet. Other matching packets are dropped.

End.DT4 vrftable TABLEID - Decapsulate the inner IPv4 packet and forward it according to the specified lookup table. *TABLEID* is either a number or a string from the file */etc/iproute2/rt_tables*. The argument must be a VRF device associated with the table id. Moreover, the VRF table associated with the table id must be configured with the VRF strict mode turned on (net.vrf.strict_mode=1). This action only accepts packets with either a zero Segments Left value or no SRH at all, and an inner IPv4 packet. Other matching packets are dropped.

End.DT46 vrftable TABLEID - Decapsulate the inner IPv4 or IPv6 packet and forward it according to the specified lookup table. *TABLEID* is either a number or a string from the file */etc/iproute2/rt_tables*. The argument must be a VRF device associated with the table id. Moreover, the VRF table associated with the table id must be configured with the VRF strict mode turned on (net.vrf.strict_mode=1). This action only accepts packets with either a zero Segments Left value or no SRH at all, and an inner IPv4 or IPv6 packet. Other matching packets are dropped.

End.B6 srh segs SEGMENTS [hmac KEYID] - Insert the specified SRH immediately after the IPv6 header, update the DA with the first segment of the newly inserted SRH, then forward the resulting packet. The original SRH is not modified. This action only accepts packets with a non-zero Segments Left value. Other matching packets are dropped.

End.B6.Encaps srh segs SEGMENTS [hmac KEYID] - Regular SRv6 processing as intermediate segment endpoint. Additionally, encapsulate the matching packet within an outer IPv6 header followed by the specified SRH. The destination address of the outer IPv6 header is set to the first segment of the new SRH. The source address is set as described in **ip-sr(8)**.

ioam6

IOAM6_TRACE_TYPE - List of IOAM data required in the trace, represented by a bitfield (24 bits).

IOAM6_NAMESPACE - Numerical value to represent an IOAM namespace. See **ip-ioam(8)**.

IOAM6_TRACE_SIZE - Size, in octets, of the pre-allocated trace data block.

expires TIME (Linux 4.4+ only)

the route will be deleted after the expires time. **Only** support IPv6 at present.

ttl-propagate { enabled | disabled }

Control whether TTL should be propagated from any encap into the un-encapsulated packet, overriding any global configuration. Only supported for MPLS at present.

ip route delete

delete route

ip route del has the same arguments as **ip route add**, but their semantics are a bit different.

Key values (**to**, **tos**, **preference** and **table**) select the route to delete. If optional attributes are present, **ip** verifies that they coincide with the attributes of the route to delete. If no route with the given key and attributes was found, **ip route del** fails.

ip route show

list routes

the command displays the contents of the routing tables or the route(s) selected by some criteria.

to SELECTOR (default)

only select routes from the given range of destinations. *SELECTOR* consists of an optional modifier (**root**, **match** or **exact**) and a prefix. **root** *PREFIX* selects routes with prefixes not shorter than *PREFIX*. F.e. **root** *0/0* selects the entire routing table. **match** *PREFIX* selects routes with prefixes not longer than *PREFIX*. F.e. **match** *10.0/16* selects *10.0/16*, *10/8* and *0/0*, but it does not select *10.1/16* and *10.0.0/24*. And **exact** *PREFIX* (or just *PREFIX*) selects routes with this exact prefix. If neither of these options are present, **ip** assumes **root** *0/0* i.e. it lists the entire table.

tos TOS**dsfield TOS**

only select routes with the given TOS.

table TABLEID

show the routes from this table(s). The default setting is to show table **main**. *TABLEID* may either be the ID of a real table or one of the special values:

all - list all of the tables.

cache - dump the routing cache.

vrf *NAME*

show the routes for the table associated with the vrf name

cloned

cached list cloned routes i.e. routes which were dynamically forked from other routes because some route attribute (f.e. MTU) was updated. Actually, it is equivalent to **table cache**.

from *SELECTOR*

the same syntax as for **to**, but it binds the source address range rather than destinations.
Note that the **from** option only works with cloned routes.

protocol *RTPROTO*

only list routes of this protocol.

scope *SCOPE_VAL*

only list routes with this scope.

type *TYPE*

only list routes of this type.

dev *NAME*

only list routes going via this device.

via [*FAMILY*] *PREFIX*

only list routes going via the nexthop routers selected by *PREFIX*.

src *PREFIX*

only list routes with preferred source addresses selected by *PREFIX*.

realm *REALMID*

realms *FROMREALM/TOREALM*

only list routes with these realms.

ip route flush

flush routing tables

this command flushes routes selected by some criteria.

The arguments have the same syntax and semantics as the arguments of **ip route show**, but routing tables are not listed but purged. The only difference is the default action: **show** dumps all the IP main routing table but **flush** prints the helper page.

With the **-statistics** option, the command becomes verbose. It prints out the number of deleted routes and the number of rounds made to flush the routing table. If the option is given twice, **ip route flush** also dumps all the deleted routes in the format described in the previous subsection.

ip route get

get a single route

this command gets a single route to a destination and prints its contents exactly as the kernel sees it.

fibmatch

Return full fib lookup matched route. Default is to return the resolved dst entry

to ADDRESS (default)

the destination address.

from ADDRESS

the source address.

tos TOS

dsfield TOS

the Type Of Service.

iif NAME

the device from which this packet is expected to arrive.

oif NAME

force the output device on which this packet will be routed.

mark MARK

the firewall mark (**fwmark**)

vrf NAME

force the vrf device on which this packet will be routed.

ipproto PROTOCOL

ip protocol as seen by the route lookup

sport NUMBER

source port as seen by the route lookup

dport NUMBER

destination port as seen by the route lookup

connected

if no source address (option **from**) was given, relookup the route with the source set to the preferred address received from the first lookup. If policy routing is used, it may be a different route.

Note that this operation is not equivalent to **ip route show**. **show** shows existing routes. **get** resolves them and creates new clones if necessary. Essentially, **get** is equivalent to sending a packet along this path. If the **iif** argument is not given, the kernel creates a route to output packets towards the requested destination. This is equivalent to pinging the destination with a subsequent **ip route ls cache**, however, no packets are actually sent. With the **iif** argument, the kernel pretends

that a packet arrived from this interface and searches for a path to forward the packet.

ip route save

save routing table information to stdout

This command behaves like **ip route show** except that the output is raw data suitable for passing to **ip route restore**.

ip route restore

restore routing table information from stdin

This command expects to read a data stream as returned from **ip route save**. It will attempt to restore the routing table information exactly as it was at the time of the save, so any translation of information in the stream (such as device indexes) must be done first. Any existing routes are left unchanged. Any routes specified in the data stream that already exist in the table will be ignored.

NOTES

Starting with Linux kernel version 3.6, there is no routing cache for IPv4 anymore. Hence **ip route show cached** will never print any entries on systems with this or newer kernel versions.

EXAMPLES**ip ro**

Show all route entries in the kernel.

ip route add default via 192.168.1.1 dev eth0

Adds a default route (for all addresses) via the local gateway 192.168.1.1 that can be reached on device eth0.

ip route add 10.1.1.0/30 encaps mpls 200/300 via 10.1.1.1 dev eth0

Adds an ipv4 route with mpls encapsulation attributes attached to it.

ip -6 route add 2001:db8:1::/64 encaps seg6 mode encaps segs 2001:db8:42::1,2001:db8:ffff::2 dev eth0

Adds an IPv6 route with SRv6 encapsulation and two segments attached.

ip -6 route add 2001:db8:1::/64 encaps seg6local action End.DT46 vrftable 100 dev vrf100

Adds an IPv6 route with SRv6 decapsulation and forward with lookup in VRF table.

ip -6 route add 2001:db8:1::/64 encaps ioam6 trace prealloc type 0x800000 ns 1 size 12 dev eth0

Adds an IPv6 route with an IOAM Pre-allocated Trace encapsulation that only includes the hop limit and the node id, configured for the IOAM namespace 1 and a pre-allocated data block of 12 octets.

ip route add 10.1.1.0/30 nhid 10

Adds an ipv4 route using nexthop object with id 10.

SEE ALSO**ip(8)****AUTHOR**

Original Manpage by Michail Litvak <mci@owl.openwall.com>

NAME

ip – Linux IPv4 protocol implementation

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h> /* superset of previous */

tcp_socket = socket(AF_INET, SOCK_STREAM, 0);
udp_socket = socket(AF_INET, SOCK_DGRAM, 0);
raw_socket = socket(AF_INET, SOCK_RAW, protocol);
```

DESCRIPTION

Linux implements the Internet Protocol, version 4, described in RFC 791 and RFC 1122. **ip** contains a level 2 multicasting implementation conforming to RFC 1112. It also contains an IP router including a packet filter.

The programming interface is BSD-sockets compatible. For more information on sockets, see **socket(7)**.

An IP socket is created using **socket(2)**:

```
socket(AF_INET, socket_type, protocol);
```

Valid socket types include **SOCK_STREAM** to open a stream socket, **SOCK_DGRAM** to open a datagram socket, and **SOCK_RAW** to open a **raw(7)** socket to access the IP protocol directly.

protocol is the IP protocol in the IP header to be received or sent. Valid values for *protocol* include:

- 0 and **IPPROTO_TCP** for **tcp(7)** stream sockets;
- 0 and **IPPROTO_UDP** for **udp(7)** datagram sockets;
- **IPPROTO_SCTP** for **sctp(7)** stream sockets; and
- **IPPROTO_UDPLITE** for **udplite(7)** datagram sockets.

For **SOCK_RAW** you may specify a valid IANA IP protocol defined in RFC 1700 assigned numbers.

When a process wants to receive new incoming packets or connections, it should bind a socket to a local interface address using **bind(2)**. In this case, only one IP socket may be bound to any given local (address, port) pair. When **IN_ADDR_ANY** is specified in the bind call, the socket will be bound to *all* local interfaces. When **listen(2)** is called on an unbound socket, the socket is automatically bound to a random free port with the local address set to **INADDR_ANY**. When **connect(2)** is called on an unbound socket, the socket is automatically bound to a random free port or to a usable shared port with the local address set to **INADDR_ANY**.

A TCP local socket address that has been bound is unavailable for some time after closing, unless the **SO_REUSEADDR** flag has been set. Care should be taken when using this flag as it makes TCP less reliable.

Address format

An IP socket address is defined as a combination of an IP interface address and a 16-bit port number. The basic IP protocol does not supply port numbers, they are implemented by higher level protocols like **udp(7)** and **tcp(7)**. On raw sockets *sin_port* is set to the IP protocol.

```
struct sockaddr_in {
    sa_family_t    sin_family; /* address family: AF_INET */
    in_port_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;  /* internet address */
};

/* Internet address */
struct in_addr {
    uint32_t        s_addr;    /* address in network byte order */
};
```

sin_family is always set to **AF_INET**. This is required; in Linux 2.2 most networking functions return **EINVAL** when this setting is missing. *sin_port* contains the port in network byte order. The port numbers below 1024 are called *privileged ports* (or sometimes: *reserved ports*). Only a privileged process (on Linux: a process that has the **CAP_NET_BIND_SERVICE** capability in the user namespace governing its network namespace) may **bind**(2) to these sockets. Note that the raw IPv4 protocol as such has no concept of a port, they are implemented only by higher protocols like **tcp**(7) and **udp**(7).

sin_addr is the IP host address. The *s_addr* member of *struct in_addr* contains the host interface address in network byte order. *in_addr* should be assigned one of the **INADDR_*** values (e.g., **INADDR_LOOPBACK**) using **htonl**(3) or set using the **inet_aton**(3), **inet_addr**(3), **inet_makeaddr**(3) library functions or directly with the name resolver (see **gethostbyname**(3)).

IPv4 addresses are divided into unicast, broadcast, and multicast addresses. Unicast addresses specify a single interface of a host, broadcast addresses specify all hosts on a network, and multicast addresses address all hosts in a multicast group. Datagrams to broadcast addresses can be sent or received only when the **SO_BROADCAST** socket flag is set. In the current implementation, connection-oriented sockets are allowed to use only unicast addresses.

Note that the address and the port are always stored in network byte order. In particular, this means that you need to call **htonl**(3) on the number that is assigned to a port. All address/port manipulation functions in the standard library work in network byte order.

There are several special addresses: **INADDR_LOOPBACK** (127.0.0.1) always refers to the local host via the loopback device; **INADDR_ANY** (0.0.0.0) means any address for binding; **INADDR_BROADCAST** (255.255.255.255) means any host and has the same effect on bind as **INADDR_ANY** for historical reasons.

Socket options

IP supports some protocol-specific socket options that can be set with **setsockopt**(2) and read with **getsockopt**(2). The socket option level for IP is **IPPROTO_IP**. A boolean integer flag is zero when it is false, otherwise true.

When an invalid socket option is specified, **getsockopt**(2) and **setsockopt**(2) fail with the error **ENOPROTOOPT**.

IP_ADD_MEMBERSHIP (since Linux 1.2)

Join a multicast group. Argument is an *ip_mreqn* structure.

```
struct ip_mreqn {
    struct in_addr imr_multiaddr; /* IP multicast group
                                   address */
    struct in_addr imr_address;   /* IP address of local
                                   interface */
    int           imr_ifindex;    /* interface index */
};
```

imr_multiaddr contains the address of the multicast group the application wants to join or leave. It must be a valid multicast address (or **setsockopt**(2) fails with the error **EINVAL**). *imr_address* is the address of the local interface with which the system should join the multicast group; if it is equal to **INADDR_ANY**, an appropriate interface is chosen by the system. *imr_ifindex* is the interface index of the interface that should join/leave the *imr_multiaddr* group, or 0 to indicate any interface.

The *ip_mreqn* structure is available only since Linux 2.2. For compatibility, the old *ip_mreq* structure (present since Linux 1.2) is still supported; it differs from *ip_mreqn* only by not including the *imr_ifindex* field. (The kernel determines which structure is being passed based on the size passed in *optlen*.)

IP_ADD_MEMBERSHIP is valid only for **setsockopt**(2).

IP_ADD_SOURCE_MEMBERSHIP (since Linux 2.4.22 / Linux 2.5.68)

Join a multicast group and allow receiving data only from a specified source. Argument is an *ip_mreq_source* structure.

```
struct ip_mreq_source {
    struct in_addr imr_multiaddr; /* IP multicast group
                                   address */
    struct in_addr imr_interface; /* IP address of local
                                   interface */
    struct in_addr imr_sourceaddr; /* IP address of
                                   multicast source */
};
```

The *ip_mreq_source* structure is similar to *ip_mreqn* described under **IP_ADD_MEMBERSHIP**. The *imr_multiaddr* field contains the address of the multicast group the application wants to join or leave. The *imr_interface* field is the address of the local interface with which the system should join the multicast group. Finally, the *imr_sourceaddr* field contains the address of the source the application wants to receive data from.

This option can be used multiple times to allow receiving data from more than one source.

IP_BIND_ADDRESS_NO_PORT (since Linux 4.2)

Inform the kernel to not reserve an ephemeral port when using **bind(2)** with a port number of 0. The port will later be automatically chosen at **connect(2)** time, in a way that allows sharing a source port as long as the 4-tuple is unique.

IP_BLOCK_SOURCE (since Linux 2.4.22 / 2.5.68)

Stop receiving multicast data from a specific source in a given group. This is valid only after the application has subscribed to the multicast group using either **IP_ADD_MEMBERSHIP** or **IP_ADD_SOURCE_MEMBERSHIP**.

Argument is an *ip_mreq_source* structure as described under **IP_ADD_SOURCE_MEMBERSHIP**.

IP_DROP_MEMBERSHIP (since Linux 1.2)

Leave a multicast group. Argument is an *ip_mreqn* or *ip_mreq* structure similar to **IP_ADD_MEMBERSHIP**.

IP_DROP_SOURCE_MEMBERSHIP (since Linux 2.4.22 / 2.5.68)

Leave a source-specific group—that is, stop receiving data from a given multicast group that come from a given source. If the application has subscribed to multiple sources within the same group, data from the remaining sources will still be delivered. To stop receiving data from all sources at once, use **IP_DROP_MEMBERSHIP**.

Argument is an *ip_mreq_source* structure as described under **IP_ADD_SOURCE_MEMBERSHIP**.

IP_FREEBIND (since Linux 2.4)

If enabled, this boolean option allows binding to an IP address that is nonlocal or does not (yet) exist. This permits listening on a socket, without requiring the underlying network interface or the specified dynamic IP address to be up at the time that the application is trying to bind to it. This option is the per-socket equivalent of the *ip_nonlocal_bind /proc* interface described below.

IP_HDRINCL (since Linux 2.0)

If enabled, the user supplies an IP header in front of the user data. Valid only for **SOCK_RAW** sockets; see **raw(7)** for more information. When this flag is enabled, the values set by **IP_OPTIONS**, **IP_TTL**, and **IP_TOS** are ignored.

IP_MSFILTER (since Linux 2.4.22 / 2.5.68)

This option provides access to the advanced full-state filtering API. Argument is an *ip_msfilter* structure.

```
struct ip_msfilter {
    struct in_addr imsf_multiaddr; /* IP multicast group
                                   address */
    struct in_addr imsf_interface; /* IP address of local
```

```

        interface */
    uint32_t      imsf_fmode;      /* Filter-mode */

    uint32_t      imsf_numsorc;   /* Number of sources in
                                  the following array */
    struct in_addr imsf_slist[1]; /* Array of source
                                  addresses */
};

}

```

There are two macros, **MCAST_INCLUDE** and **MCAST_EXCLUDE**, which can be used to specify the filtering mode. Additionally, the **IP_MSFILTER_SIZE(n)** macro exists to determine how much memory is needed to store *ip_msfilter* structure with *n* sources in the source list.

For the full description of multicast source filtering refer to RFC 3376.

IP_MTU (since Linux 2.2)

Retrieve the current known path MTU of the current socket. Returns an integer.

IP_MTU is valid only for **getsockopt(2)** and can be employed only when the socket has been connected.

IP_MTU_DISCOVER (since Linux 2.2)

Set or receive the Path MTU Discovery setting for a socket. When enabled, Linux will perform Path MTU Discovery as defined in RFC 1191 on **SOCK_STREAM** sockets. For non-**SOCK_STREAM** sockets, **IP_PMTUDISC_DO** forces the don't-fragment flag to be set on all outgoing packets. It is the user's responsibility to packetize the data in MTU-sized chunks and to do the retransmits if necessary. The kernel will reject (with **EMSGSIZE**) datagrams that are bigger than the known path MTU. **IP_PMTUDISC_WANT** will fragment a datagram if needed according to the path MTU, or will set the don't-fragment flag otherwise.

The system-wide default can be toggled between **IP_PMTUDISC_WANT** and **IP_PMTUDISC_DONT** by writing (respectively, zero and nonzero values) to the */proc/sys/net/ipv4/ip_no_pmtu_disc* file.

Path MTU discovery value	Meaning
IP_PMTUDISC_WANT	Use per-route settings.
IP_PMTUDISC_DONT	Never do Path MTU Discovery.
IP_PMTUDISC_DO	Always do Path MTU Discovery.
IP_PMTUDISC_PROBE	Set DF but ignore Path MTU.

When PMTU discovery is enabled, the kernel automatically keeps track of the path MTU per destination host. When it is connected to a specific peer with **connect(2)**, the currently known path MTU can be retrieved conveniently using the **IP_MTU** socket option (e.g., after an **EMSGSIZE** error occurred). The path MTU may change over time. For connectionless sockets with many destinations, the new MTU for a given destination can also be accessed using the error queue (see **IP_RECVERR**). A new error will be queued for every incoming MTU update.

While MTU discovery is in progress, initial packets from datagram sockets may be dropped. Applications using UDP should be aware of this and not take it into account for their packet retransmit strategy.

To bootstrap the path MTU discovery process on unconnected sockets, it is possible to start with a big datagram size (headers up to 64 kilobytes long) and let it shrink by updates of the path MTU.

To get an initial estimate of the path MTU, connect a datagram socket to the destination address using **connect(2)** and retrieve the MTU by calling **getsockopt(2)** with the **IP_MTU** option.

It is possible to implement RFC 4821 MTU probing with **SOCK_DGRAM** or **SOCK_RAW** sockets by setting a value of **IP_PMTUDISC_PROBE** (available since Linux 2.6.22). This is also particularly useful for diagnostic tools such as **tracepath(8)** that wish to deliberately send probe packets larger than the observed Path MTU.

IP_MULTICAST_ALL (since Linux 2.6.31)

This option can be used to modify the delivery policy of multicast messages to sockets bound to the wildcard **INADDR_ANY** address. The argument is a boolean integer (defaults to 1). If set to 1, the socket will receive messages from all the groups that have been joined globally on the whole system. Otherwise, it will deliver messages only from the groups that have been explicitly joined (for example via the **IP_ADD_MEMBERSHIP** option) on this particular socket.

IP_MULTICAST_IF (since Linux 1.2)

Set the local device for a multicast socket. The argument for **setsockopt(2)** is an *ip_mreqn* or (since Linux 3.5) *ip_mreq* structure similar to **IP_ADD_MEMBERSHIP**, or an *in_addr* structure. (The kernel determines which structure is being passed based on the size passed in *optlen*.) For **getsockopt(2)**, the argument is an *in_addr* structure.

IP_MULTICAST_LOOP (since Linux 1.2)

Set or read a boolean integer argument that determines whether sent multicast packets should be looped back to the local sockets.

IP_MULTICAST_TTL (since Linux 1.2)

Set or read the time-to-live value of outgoing multicast packets for this socket. It is very important for multicast packets to set the smallest TTL possible. The default is 1 which means that multicast packets don't leave the local network unless the user program explicitly requests it. Argument is an integer.

IP_NODEFRAG (since Linux 2.6.36)

If enabled (argument is nonzero), the reassembly of outgoing packets is disabled in the netfilter layer. The argument is an integer.

This option is valid only for **SOCK_RAW** sockets.

IP_OPTIONS (since Linux 2.0)

Set or get the IP options to be sent with every packet from this socket. The arguments are a pointer to a memory buffer containing the options and the option length. The **setsockopt(2)** call sets the IP options associated with a socket. The maximum option size for IPv4 is 40 bytes. See RFC 791 for the allowed options. When the initial connection request packet for a **SOCK_STREAM** socket contains IP options, the IP options will be set automatically to the options from the initial packet with routing headers reversed. Incoming packets are not allowed to change options after the connection is established. The processing of all incoming source routing options is disabled by default and can be enabled by using the *accept_source_route /proc* interface. Other options like timestamps are still handled. For datagram sockets, IP options can be set only by the local user. Calling **getsockopt(2)** with **IP_OPTIONS** puts the current IP options used for sending into the supplied buffer.

IP_PASSEC (since Linux 2.6.17)

If labeled IPSEC or NetLabel is configured on the sending and receiving hosts, this option enables receiving of the security context of the peer socket in an ancillary message of type **SCM_SECURITY** retrieved using **recvmsg(2)**. This option is supported only for UDP sockets; for TCP or SCTP sockets, see the description of the **SO_PEERSEC** option below.

The value given as an argument to **setsockopt(2)** and returned as the result of **getsockopt(2)** is an integer boolean flag.

The security context returned in the **SCM_SECURITY** ancillary message is of the same format as the one described under the **SO_PEERSEC** option below.

Note: the reuse of the **SCM_SECURITY** message type for the **IP_PASSEC** socket option was likely a mistake, since other IP control messages use their own numbering scheme in the IP namespace and often use the socket option value as the message type. There is no conflict currently since the IP option with the same value as **SCM_SECURITY** is **IP_HDRINCL** and this is never used for a control message type.

IP_PKTINFO (since Linux 2.2)

Pass an **IP_PKTINFO** ancillary message that contains a *pktinfo* structure that supplies some information about the incoming packet. This works only for datagram oriented sockets. The argument is a flag that tells the socket whether the **IP_PKTINFO** message should be passed or not. The message itself can be sent/retrieved only as a control message with a packet using **recvmsg(2)** or **sendmsg(2)**.

```
struct in_pktinfo {
    unsigned int    ipi_ifindex; /* Interface index */
    struct in_addr ipi_spec_dst; /* Local address */
    struct in_addr ipi_addr;    /* Header Destination
                                address */
};
```

ipi_ifindex is the unique index of the interface the packet was received on. *ipi_spec_dst* is the local address of the packet and *ipi_addr* is the destination address in the packet header. If **IP_PKTINFO** is passed to **sendmsg(2)** and *ipi_spec_dst* is not zero, then it is used as the local source address for the routing table lookup and for setting up IP source route options. When *ipi_ifindex* is not zero, the primary local address of the interface specified by the index overwrites *ipi_spec_dst* for the routing table lookup.

IP_RECVERR (since Linux 2.2)

Enable extended reliable error message passing. When enabled on a datagram socket, all generated errors will be queued in a per-socket error queue. When the user receives an error from a socket operation, the errors can be received by calling **recvmsg(2)** with the **MSG_ERRQUEUE** flag set. The *sock_extended_err* structure describing the error will be passed in an ancillary message with the type **IP_RECVERR** and the level **IPPROTO_IP**. This is useful for reliable error handling on unconnected sockets. The received data portion of the error queue contains the error packet.

The **IP_RECVERR** control message contains a *sock_extended_err* structure:

```
#define SO_EE_ORIGIN_NONE      0
#define SO_EE_ORIGIN_LOCAL      1
#define SO_EE_ORIGIN_ICMP       2
#define SO_EE_ORIGIN_ICMP6      3

struct sock_extended_err {
    uint32_t ee_errno;        /* error number */
    uint8_t  ee_origin;       /* where the error originated */
    uint8_t  ee_type;         /* type */
    uint8_t  ee_code;         /* code */
    uint8_t  ee_pad;
    uint32_t ee_info;         /* additional information */
    uint32_t ee_data;         /* other data */
    /* More data may follow */
};

struct sockaddr *SO_EE_OFFENDER(struct sock_extended_err *);
```

ee_errno contains the *errno* number of the queued error. *ee_origin* is the origin code of where the error originated. The other fields are protocol-specific. The macro **SO_EE_OFFENDER** returns a pointer to the address of the network object where the error originated from given a pointer to the ancillary message. If this address is not known, the *sa_family* member of the *sockaddr* contains **AF_UNSPEC** and the other fields of the *sockaddr* are undefined.

IP uses the *sock_extended_err* structure as follows: *ee_origin* is set to **SO_EE_ORIGIN_ICMP** for errors received as an ICMP packet, or **SO_EE_ORIGIN_LOCAL** for locally generated errors. Unknown values should be ignored. *ee_type* and *ee_code* are set from the type and code

fields of the ICMP header. *ee_info* contains the discovered MTU for **EMSGSIZE** errors. The message also contains the *sockaddr_in* of the node caused the error, which can be accessed with the **SO_EE_OFFENDER** macro. The *sin_family* field of the **SO_EE_OFFENDER** address is **AF_UNSPEC** when the source was unknown. When the error originated from the network, all IP options (**IP_OPTIONS**, **IP_TTL**, etc.) enabled on the socket and contained in the error packet are passed as control messages. The payload of the packet causing the error is returned as normal payload. Note that TCP has no error queue; **MSG_ERRQ UEUE** is not permitted on **SOCK_STREAM** sockets. **IP_RECVERR** is valid for TCP, but all errors are returned by socket function return or **SO_ERROR** only.

For raw sockets, **IP_RECVERR** enables passing of all received ICMP errors to the application, otherwise errors are reported only on connected sockets

It sets or retrieves an integer boolean flag. **IP_RECVERR** defaults to off.

IP_RECVOPTS (since Linux 2.2)

Pass all incoming IP options to the user in a **IP_OPTIONS** control message. The routing header and other options are already filled in for the local host. Not supported for **SOCK_STREAM** sockets.

IP_RECVORIGSTADDR (since Linux 2.6.29)

This boolean option enables the **IP_ORIGSTADDR** ancillary message in **recvmsg(2)**, in which the kernel returns the original destination address of the datagram being received. The ancillary message contains a *struct sockaddr_in*.

IP_RECVTOS (since Linux 2.2)

If enabled, the **IP_TOS** ancillary message is passed with incoming packets. It contains a byte which specifies the Type of Service/Precedence field of the packet header. Expects a boolean integer flag.

IP_RECVTTL (since Linux 2.2)

When this flag is set, pass a **IP_TTL** control message with the time-to-live field of the received packet as a 32 bit integer. Not supported for **SOCK_STREAM** sockets.

IP_RECVTOPTS (since Linux 2.2)

Identical to **IP_RECVOPTS**, but returns raw unprocessed options with timestamp and route record options not filled in for this hop.

IP_ROUTER_ALERT (since Linux 2.2)

Pass all to-be forwarded packets with the IP Router Alert option set to this socket. Valid only for raw sockets. This is useful, for instance, for user-space RSVP daemons. The tapped packets are not forwarded by the kernel; it is the user's responsibility to send them out again. Socket binding is ignored, such packets are filtered only by protocol. Expects an integer flag.

IP_TOS (since Linux 1.0)

Set or receive the Type-Of-Service (TOS) field that is sent with every IP packet originating from this socket. It is used to prioritize packets on the network. TOS is a byte. There are some standard TOS flags defined: **IPTOS_LOWDELAY** to minimize delays for interactive traffic, **IP-TOS_THROUGHPUT** to optimize throughput, **IPTOS_RELIABILITY** to optimize for reliability, **IPTOS_MINCOST** should be used for "filler data" where slow transmission doesn't matter. At most one of these TOS values can be specified. Other bits are invalid and shall be cleared. Linux sends **IPTOS_LOWDELAY** datagrams first by default, but the exact behavior depends on the configured queueing discipline. Some high-priority levels may require superuser privileges (the **CAP_NET_ADMIN** capability).

IP_TRANSPARENT (since Linux 2.6.24)

Setting this boolean option enables transparent proxying on this socket. This socket option allows the calling application to bind to a nonlocal IP address and operate both as a client and a server with the foreign address as the local endpoint. NOTE: this requires that routing be set up in a way that packets going to the foreign address are routed through the TProxy box (i.e., the system

hosting the application that employs the **IP_TRANSPARENT** socket option). Enabling this socket option requires superuser privileges (the **CAP_NET_ADMIN** capability).

TProxy redirection with the iptables TPROXY target also requires that this option be set on the redirected socket.

IP_TTL (since Linux 1.0)

Set or retrieve the current time-to-live field that is used in every packet sent from this socket.

IP_UNBLOCK_SOURCE (since Linux 2.4.22 / 2.5.68)

Unblock previously blocked multicast source. Returns **EADDRNOTAVAIL** when given source is not being blocked.

Argument is an *ip_mreq_source* structure as described under **IP_ADD_SOURCE_MEMBERSHIP**.

SO_PEERSEC (since Linux 2.6.17)

If labeled IPSEC or NetLabel is configured on both the sending and receiving hosts, this read-only socket option returns the security context of the peer socket connected to this socket. By default, this will be the same as the security context of the process that created the peer socket unless overridden by the policy or by a process with the required permissions.

The argument to **getsockopt(2)** is a pointer to a buffer of the specified length in bytes into which the security context string will be copied. If the buffer length is less than the length of the security context string, then **getsockopt(2)** returns -1, sets *errno* to **ERANGE**, and returns the required length via *optlen*. The caller should allocate at least **NAME_MAX** bytes for the buffer initially, although this is not guaranteed to be sufficient. Resizing the buffer to the returned length and retrying may be necessary.

The security context string may include a terminating null character in the returned length, but is not guaranteed to do so: a security context "foo" might be represented as either {'f','o','o'} of length 3 or {'f','o','o','\0'} of length 4, which are considered to be interchangeable. The string is printable, does not contain non-terminating null characters, and is in an unspecified encoding (in particular, it is not guaranteed to be ASCII or UTF-8).

The use of this option for sockets in the **AF_INET** address family is supported since Linux 2.6.17 for TCP sockets, and since Linux 4.17 for SCTP sockets.

For SELinux, NetLabel conveys only the MLS portion of the security context of the peer across the wire, defaulting the rest of the security context to the values defined in the policy for the netmsg initial security identifier (SID). However, NetLabel can be configured to pass full security contexts over loopback. Labeled IPSEC always passes full security contexts as part of establishing the security association (SA) and looks them up based on the association for each packet.

/proc interfaces

The IP protocol supports a set of */proc* interfaces to configure some global parameters. The parameters can be accessed by reading or writing files in the directory */proc/sys/net/ipv4/*. Interfaces described as *Boolean* take an integer value, with a nonzero value ("true") meaning that the corresponding option is enabled, and a zero value ("false") meaning that the option is disabled.

ip_always_defrag (Boolean; since Linux 2.2.13)

[New with Linux 2.2.13; in earlier kernel versions this feature was controlled at compile time by the **CONFIG_IP_ALWAYS_DEFRAG** option; this option is not present in Linux 2.4.x and later]

When this boolean flag is enabled (not equal 0), incoming fragments (parts of IP packets that arose when some host between origin and destination decided that the packets were too large and cut them into pieces) will be reassembled (defragmented) before being processed, even if they are about to be forwarded.

Enable only if running either a firewall that is the sole link to your network or a transparent proxy; never ever use it for a normal router or host. Otherwise, fragmented communication can be disturbed if the fragments travel over different links. Defragmentation also has a large memory and

CPU time cost.

This is automatically turned on when masquerading or transparent proxying are configured.

ip_autoconfig (since Linux 2.2 to Linux 2.6.17)

Not documented.

ip_default_ttl (integer; default: 64; since Linux 2.2)

Set the default time-to-live value of outgoing packets. This can be changed per socket with the **IP_TTL** option.

ip_dynaddr (Boolean; default: disabled; since Linux 2.0.31)

Enable dynamic socket address and masquerading entry rewriting on interface address change. This is useful for dialup interface with changing IP addresses. 0 means no rewriting, 1 turns it on and 2 enables verbose mode.

ip_forward (Boolean; default: disabled; since Linux 1.2)

Enable IP forwarding with a boolean flag. IP forwarding can be also set on a per-interface basis.

ip_local_port_range (since Linux 2.2)

This file contains two integers that define the default local port range allocated to sockets that are not explicitly bound to a port number—that is, the range used for *ephemeral ports*. An ephemeral port is allocated to a socket in the following circumstances:

- the port number in a socket address is specified as 0 when calling **bind**(2);
- **listen**(2) is called on a stream socket that was not previously bound;
- **connect**(2) was called on a socket that was not previously bound;
- **sendto**(2) is called on a datagram socket that was not previously bound.

Allocation of ephemeral ports starts with the first number in *ip_local_port_range* and ends with the second number. If the range of ephemeral ports is exhausted, then the relevant system call returns an error (but see BUGS).

Note that the port range in *ip_local_port_range* should not conflict with the ports used by masquerading (although the case is handled). Also, arbitrary choices may cause problems with some firewall packet filters that make assumptions about the local ports in use. The first number should be at least greater than 1024, or better, greater than 4096, to avoid clashes with well known ports and to minimize firewall problems.

ip_no_pmtu_disc (Boolean; default: disabled; since Linux 2.2)

If enabled, don't do Path MTU Discovery for TCP sockets by default. Path MTU discovery may fail if misconfigured firewalls (that drop all ICMP packets) or misconfigured interfaces (e.g., a point-to-point link where the both ends don't agree on the MTU) are on the path. It is better to fix the broken routers on the path than to turn off Path MTU Discovery globally, because not doing it incurs a high cost to the network.

ip_nonlocal_bind (Boolean; default: disabled; since Linux 2.4)

If set, allows processes to **bind**(2) to nonlocal IP addresses, which can be quite useful, but may break some applications.

ip6frag_time (integer; default: 30)

Time in seconds to keep an IPv6 fragment in memory.

ip6frag_secret_interval (integer; default: 600)

Regeneration interval (in seconds) of the hash secret (or lifetime for the hash secret) for IPv6 fragments.

ipfrag_high_thresh (integer), *ipfrag_low_thresh* (integer)

If the amount of queued IP fragments reaches *ipfrag_high_thresh*, the queue is pruned down to *ipfrag_low_thresh*. Contains an integer with the number of bytes.

*neigh/** See **arp(7)**.

Ioctl

All ioctls described in **socket(7)** apply to **ip**.

Ioctl to configure generic device parameters are described in **netdevice(7)**.

ERRORS

EACCES

The user tried to execute an operation without the necessary permissions. These include: sending a packet to a broadcast address without having the **SO_BROADCAST** flag set; sending a packet via a *prohibit* route; modifying firewall settings without superuser privileges (the **CAP_NET_ADMIN** capability); binding to a privileged port without superuser privileges (the **CAP_NET_BIND_SERVICE** capability).

EADDRINUSE

Tried to bind to an address already in use.

EADDRNOTAVAIL

A nonexistent interface was requested or the requested source address was not local.

EAGAIN

Operation on a nonblocking socket would block.

EALREADY

A connection operation on a nonblocking socket is already in progress.

ECONNABORTED

A connection was closed during an **accept(2)**.

EHOSTUNREACH

No valid routing table entry matches the destination address. This error can be caused by an ICMP message from a remote router or for the local routing table.

EINVAL

Invalid argument passed. For send operations this can be caused by sending to a *blackhole* route.

EISCONN

connect(2) was called on an already connected socket.

EMSGSIZE

Datagram is bigger than an MTU on the path and it cannot be fragmented.

ENOBUFS, ENOMEM

Not enough free memory. This often means that the memory allocation is limited by the socket buffer limits, not by the system memory, but this is not 100% consistent.

ENOENT

SIOCGSTAMP was called on a socket where no packet arrived.

ENOPKG

A kernel subsystem was not configured.

ENOPROTOOPT and EOPNOTSUPP

Invalid socket option passed.

ENOTCONN

The operation is defined only on a connected socket, but the socket wasn't connected.

EPERM

User doesn't have permission to set high priority, change configuration, or send signals to the requested process or group.

EPIPE The connection was unexpectedly closed or shut down by the other end.

ESOCKTNOSUPPORT

The socket is not configured or an unknown socket type was requested.

Other errors may be generated by the overlaying protocols; see **tcp(7)**, **raw(7)**, **udp(7)**, and **socket(7)**.

NOTES

IP_FREEBIND, **IP_MSFILTER**, **IP_MTU**, **IP_MTU_DISCOVER**, **IP_RECVORIGDSTADDR**, **IP_PASSEC**, **IP_PKTINFO**, **IP_RECVERR**, **IP_ROUTER_ALERT**, and **IP_TRANSPARENT** are Linux-specific.

Be very careful with the **SO_BROADCAST** option – it is not privileged in Linux. It is easy to overload the network with careless broadcasts. For new application protocols it is better to use a multicast group instead of broadcasting. Broadcasting is discouraged.

Some other BSD sockets implementations provide **IP_RCVDSTADDR** and **IP_RECVIF** socket options to get the destination address and the interface of received datagrams. Linux has the more general **IP_PKTINFO** for the same task.

Some BSD sockets implementations also provide an **IP_RECVTTL** option, but an ancillary message with type **IP_RECVTTL** is passed with the incoming packet. This is different from the **IP_TTL** option used in Linux.

Using the **SOL_IP** socket options level isn't portable; BSD-based stacks use the **IPPROTO_IP** level.

INADDR_ANY (0.0.0.0) and **INADDR_BROADCAST** (255.255.255.255) are byte-order-neutral. This means **htonl(3)** has no effect on them.

Compatibility

For compatibility with Linux 2.0, the obsolete **socket(AF_INET, SOCK_PACKET, protocol)** syntax is still supported to open a **packet(7)** socket. This is deprecated and should be replaced by **socket(AF_PACKET, SOCK_RAW, protocol)** instead. The main difference is the new **sockaddr_ll** address structure for generic link layer information instead of the old **sockaddr_pkt**.

BUGS

There are too many inconsistent error values.

The error used to diagnose exhaustion of the ephemeral port range differs across the various system calls (**connect(2)**, **bind(2)**, **listen(2)**, **sendto(2)**) that can assign ephemeral ports.

The ioctls to configure IP-specific interface options and ARP tables are not described.

Receiving the original destination address with **MSG_ERRQUEUE** in *msg_name* by **recvmsg(2)** does not work in some Linux 2.2 kernels.

SEE ALSO

recvmsg(2), **sendmsg(2)**, **byteorder(3)**, **capabilities(7)**, **icmp(7)**, **ipv6(7)**, **netdevice(7)**, **netlink(7)**, **raw(7)**, **socket(7)**, **tcp(7)**, **udp(7)**, **ip(8)**

The kernel source file *Documentation/networking/ip-sysctl.txt*.

RFC 791 for the original IP specification. RFC 1122 for the IPv4 host requirements. RFC 1812 for the IPv4 router requirements.

NAME

ip – show / manipulate routing, network devices, interfaces and tunnels

SYNOPSIS

ip [*OPTIONS*] *OBJECT* { *COMMAND* | **help** }

ip [**-force**] **-batch** *filename*

OBJECT := { **link** | **address** | **addrlabel** | **route** | **rule** | **neigh** | **ntable** | **tunnel** | **tuntap** | **maddress** |
mroute | **mrule** | **monitor** | **xfrm** | **netns** | **l2tp** | **tcp_metrics** | **token** | **macsec** | **vrf** |
mptcp | **ioam** }

OPTIONS := { **-V[ersion]** | **-h[uman-readable]** | **-s[tatistics]** | **-d[etails]** | **-r[esolve]** | **-iec** | **-f[amily]** | {
inet | **inet6** | **link** } | **-4** | **-6** | **-B** | **-0** | **-l[oops]** { **maximum-addr-flush-attempts** } |
-o[neline] | **-rc[vbuf]** [**size**] | **-t[imestamp]** | **-ts[hort]** | **-n[etns]** *name* | **-N[umeric]** |
-a[ll] | **-c[olor]** | **-br[ief]** | **-j[son]** | **-p[retty]** }

OPTIONS**-V, -Version**

Print the version of the **ip** utility and exit.

-h, -human, -human-readable

output statistics with human readable values followed by suffix.

-b, -batch <FILENAME>

Read commands from provided file or standard input and invoke them. First failure will cause termination of ip.

-force Don't terminate ip on errors in batch mode. If there were any errors during execution of the commands, the application return code will be non zero.

-s, -stats, -statistics

Output more information. If the option appears twice or more, the amount of information increases. As a rule, the information is statistics or some time values.

-d, -details

Output more detailed information.

-l, -loops <COUNT>

Specify maximum number of loops the 'ip address flush' logic will attempt before giving up. The default is 10. Zero (0) means loop until all addresses are removed.

-f, -family <FAMILY>

Specifies the protocol family to use. The protocol family identifier can be one of **inet**, **inet6**, **bridge**, **mpls** or **link**. If this option is not present, the protocol family is guessed from other arguments. If the rest of the command line does not give enough information to guess the family, **ip** falls back to the default one, usually **inet** or **any**. **link** is a special family identifier meaning that no networking protocol is involved.

-4 shortcut for **-family inet**.

-6 shortcut for **-family inet6**.

-B shortcut for **-family bridge**.

-M shortcut for **-family mpls**.

-0 shortcut for **-family link**.

-o, -oneline

output each record on a single line, replacing line feeds with the '\` character. This is convenient when you want to count records with **wc(1)** or to **grep(1)** the output.

-r, -resolve

use the system's name resolver to print DNS names instead of host addresses.

-n, -netns <NETNS>

switches **ip** to the specified network namespace *NETNS*. Actually it just simplifies executing of:

```
ip netns exec NETNS ip [ OPTIONS ] OBJECT { COMMAND | help }
```

to

```
ip -n[etns] NETNS [ OPTIONS ] OBJECT { COMMAND | help }
```

-N, -Numeric

Print the number of protocol, scope, dsfield, etc directly instead of converting it to human readable name.

-a, -all

executes specified command over all objects, it depends if command supports this option.

-c[color][={always|auto|never}]

Configure color output. If parameter is omitted or **always**, color output is enabled regardless of stdout state. If parameter is **auto**, stdout is checked to be a terminal before enabling color output. If parameter is **never**, color output is disabled. If specified multiple times, the last one takes precedence. This flag is ignored if **-json** is also given.

Used color palette can be influenced by **COLORFGBG** environment variable (see **ENVIRONMENT**).

-t, -timestamp

display current time when using monitor option.

-ts, -tshort

Like **-timestamp**, but use shorter format.

-rc, -recvbuf<SIZE>

Set the netlink socket receive buffer size, defaults to 1MB.

-iec print human readable rates in IEC units (e.g. 1Ki = 1024).**-br, -brief**

Print only basic information in a tabular format for better readability. This option is currently only supported by **ip addr show**, **ip link show** & **ip neigh show** commands.

-j, -json

Output results in JavaScript Object Notation (JSON).

-p, -pretty

The default JSON format is compact and more efficient to parse but hard for most users to read.
This flag adds indentation for readability.

IP - COMMAND SYNTAX*OBJECT***address**

- protocol (IP or IPv6) address on a device.

addrlabel

- label configuration for protocol address selection.

ioam - manage IOAM namespaces and IOAM schemas.**l2tp** - tunnel ethernet over IP (L2TPv3).**link** - network device.**maddress**

- multicast address.

monitor

- watch for netlink messages.

mptcp - manage MPTCP path manager.**mroute** - multicast routing cache entry.**mrule** - rule in multicast routing policy database.**neighbour**

- manage ARP or NDISC cache entries.

netns - manage network namespaces.

ntable - manage the neighbor cache's operation.

route - routing table entry.

rule - rule in routing policy database.

tcp_metrics/tcpmetrics

- manage TCP Metrics

token - manage tokenized interface identifiers.

tunnel - tunnel over IP.

tuntap - manage TUN/TAP devices.

vrf - manage virtual routing and forwarding devices.

xfrm - manage IPSec policies.

The names of all objects may be written in full or abbreviated form, for example **address** can be abbreviated as **addr** or just **a**.

COMMAND

Specifies the action to perform on the object. The set of possible actions depends on the object type. As a rule, it is possible to **add**, **delete** and **show** (or **list**) objects, but some objects do not allow all of these operations or have some additional commands. The **help** command is available for all objects. It prints out a list of available commands and argument syntax conventions.

If no command is given, some default command is assumed. Usually it is **list** or, if the objects of this class cannot be listed, **help**.

ENVIRONMENT

COLORFGBG

If set, its value is used for detection whether background is dark or light and use contrast colors for it.

COLORFGBG environment variable usually contains either two or three values separated by semicolons; we want the last value in either case. If this value is 0-6 or 8, chose colors suitable for dark background:

```
COLORFGBG=";0" ip -c a
```

EXIT STATUS

Exit status is 0 if command was successful, and 1 if there is a syntax error. If an error was reported by the kernel exit status is 2.

EXAMPLES

`ip addr`

Shows addresses assigned to all network interfaces.

ip neigh
Shows the current neighbour table in kernel.

ip link set x up
Bring up interface x.

ip link set x down
Bring down interface x.

ip route
Show table routes.

HISTORY

ip was written by Alexey N. Kuznetsov and added in Linux 2.2.

SEE ALSO

ip-address(8), ip-addrlabel(8), ip-ioam(8), ip-l2tp(8), ip-link(8), ip-maddress(8), ip-monitor(8), ip-mptcp(8), ip-mroute(8), ip-neighbour(8), ip-netns(8), ip-ntable(8), ip-route(8), ip-rule(8), ip-tcp_metrics(8), ip-token(8), ip-tunnel(8), ip-vrf(8), ip-xfrm(8)
IP Command reference **ip-cref.ps**

REPORTING BUGS

Report any bugs to the Network Developers mailing list <netdev@vger.kernel.org> where the development and maintenance is primarily done. You do not have to be subscribed to the list to send a message there.

AUTHOR

Original Manpage by Michail Litvak <mci@owl.openwall.com>

NAME

xtables-nft — iptables using nftables kernel api

DESCRIPTION

xtables-nft are versions of iptables that use the nftables API. This is a set of tools to help the system administrator migrate the ruleset from **iptables(8)**, **ip6tables(8)**, **arptables(8)**, and **ebtables(8)** to **nftables(8)**.

The **xtables-nft** set is composed of several commands:

- **iptables-nft**
- **iptables-nft-save**
- **iptables-nft-restore**
- **ip6tables-nft**
- **ip6tables-nft-save**
- **ip6tables-nft-restore**
- **arptables-nft**
- **ebtables-nft**

These tools use the libxtables framework extensions and hook to the nf_tables kernel subsystem using the **nft_compat** module.

USAGE

The xtables-nft tools allow you to manage the nf_tables backend using the native syntax of **iptables(8)**, **ip6tables(8)**, **arptables(8)**, and **ebtables(8)**.

You should use the xtables-nft tools exactly the same way as you would use the corresponding original tools.

Adding a rule will result in that rule being added to the nf_tables kernel subsystem instead. Listing the ruleset will use the nf_tables backend as well.

When these tools were designed, the main idea was to replace each legacy binary with a symlink to the xtables-nft program, for example:

```
/sbin/iptables -> /usr/sbin/iptables-nft-multi
/sbin/ip6tables -> /usr/sbin/ip6tables-nft-multi
/sbin/arptables -> /usr/sbin/arptables-nft-multi
/sbin/ebtables -> /usr/sbin/ebtables-nft-multi
```

The iptables version string will indicate whether the legacy API (get/setsockopt) or the new nf_tables api is used:

```
iptables -V
iptables v1.7 (nf_tables)
```

DIFFERENCES TO LEGACY IPTABLES

Because the xtables-nft tools use the nf_tables kernel API, rule additions and deletions are always atomic. Unlike iptables-legacy, iptables-nft -A ... will NOT need to retrieve the current ruleset from the kernel, change it, and re-load the altered ruleset. Instead, iptables-nft will tell the kernel to add one rule. For this reason, the iptables-legacy --wait option is a no-op in iptables-nft.

Use of the xtables-nft tools allow monitoring ruleset changes using the **xtables-monitor(8)** command.

When using `-j TRACE` to debug packet traversal to the ruleset, note that you will need to use **xtables-monitor(8)** in `--trace` mode to obtain monitoring trace events.

EXAMPLES

One basic example is creating the skeleton ruleset in nf_tables from the xtables-nft tools, in a fresh machine:

```
root@machine:~# iptables-nft -L
[...]
root@machine:~# ip6tables-nft -L
[...]
root@machine:~# arptables-nft -L
[...]
root@machine:~# ebtables-nft -L
[...]
root@machine:~# nft list ruleset
table ip filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}
table ip6 filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}
table bridge filter {
    chain INPUT {
        type filter hook input priority -200; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority -200; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority -200; policy accept;
    }
}
```

```

}
table arp filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}

```

(please note that in fresh machines, listing the ruleset for the first time results in all tables an chain being created).

To migrate your complete filter ruleset, in the case of **iptables(8)**, you would use:

```

root@machine:~# iptables-legacy-save > myruleset # reads from x_tables
root@machine:~# iptables-nft-restore myruleset # writes to nf_tables
or
root@machine:~# iptables-legacy-save | iptables-translate-restore | less

```

to see how rules would look like in the nft **nft(8)** syntax.

LIMITATIONS

You should use **Linux kernel >= 4.17**.

The CLUSTERIP target is not supported.

To get up-to-date information about this, please head to <http://wiki.nftables.org/>.

SEE ALSO

nft(8), xtables-translate(8), xtables-monitor(8)

AUTHORS

The nftables framework is written by the Netfilter project (<https://www.netfilter.org>).

This manual page was written by Arturo Borrero Gonzalez <arturo@debian.org> for the Debian project, but may be used by others.

This documentation is free/libre under the terms of the GPLv2+.

NAME

iptables-save — dump iptables rules
ip6tables-save — dump ip6tables rules

SYNOPSIS

```
iptables-save [-M modprobe] [-c] [-t table] [-f filename]  
ip6tables-save [-M modprobe] [-c] [-t table] [-f filename]
```

DESCRIPTION

iptables-save and **ip6tables-save** are used to dump the contents of IP or IPv6 Table in easily parseable format either to STDOUT or to a specified file.

-M, --modprobe *modprobe_program*

Specify the path to the modprobe program. By default, **iptables-save** will inspect /proc/sys/kernel/modprobe to determine the executable's path.

-f, --file *filename*

Specify a filename to log the output to. If not specified, **iptables-save** will log to STDOUT.

-c, --counters

include the current values of all packet and byte counters in the output

-t, --table *tablename*

restrict output to only one table. If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there.

If not specified, output includes all available tables.

BUGS

None known as of iptables-1.2.1 release

AUTHORS

Harald Welte <laforg@gnumonks.org>
Rusty Russell <rusty@rustcorp.com.au>
Andras Kis-Szabo <kisza@sch.bme.hu> contributed ip6tables-save.

SEE ALSO

iptables-apply(8),iptables-restore(8), iptables(8)

The **iptables**-HOWTO, which details more **iptables** usage, the **NAT**-HOWTO, which details **NAT**, and the **netfilter-hacking**-HOWTO which details the internals.

NAME

iptables/**ip6tables** — administration tool for IPv4/IPv6 packet filtering and NAT

SYNOPSIS

```
iptables [-t table] {-A|-C|-D} chain rule-specification
ip6tables [-t table] {-A|-C|-D} chain rule-specification
iptables [-t table] -I chain [rulenumber] rule-specification
iptables [-t table] -R chain rulenumber rule-specification
iptables [-t table] -D chain rulenumber
iptables [-t table] -S [chain [rulenumber]]
iptables [-t table] {-F|-L|-Z} [chain [rulenumber]] [options...]
iptables [-t table] -N chain
iptables [-t table] -X [chain]
iptables [-t table] -P chain target
iptables [-t table] -E old-chain-name new-chain-name
rule-specification = [matches...] [target]
match = -m matchname [per-match-options]
target = -j targetname [per-target-options]
```

DESCRIPTION

Iptables and **ip6tables** are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a ‘target’, which may be a jump to a user-defined chain in the same table.

TARGETS

A firewall rule specifies criteria for a packet and a target. If the packet does not match, the next rule in the chain is examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain, one of the targets described in **iptables-extensions(8)**, or one of the special values **ACCEPT**, **DROP** or **RETURN**.

ACCEPT means to let the packet through. **DROP** means to drop the packet on the floor. **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target **RETURN** is matched, the target specified by the chain policy determines the fate of the packet.

TABLES

There are currently five independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

-t, --table table

This option specifies the packet matching table which the command should operate on. If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there.

The tables are as follows:

filter: This is the default table (if no **-t** option is passed). It contains the built-in chains **INPUT** (for packets destined to local sockets), **FORWARD** (for packets being routed through the box), and **OUTPUT** (for locally-generated packets).

nat: This table is consulted when a packet that creates a new connection is encountered. It consists of four built-ins: **PREROUTING** (for altering packets as soon as they come in),

INPUT (for altering packets destined for local sockets), **OUTPUT** (for altering locally-generated packets before routing), and **POSTROUTING** (for altering packets as they are about to go out). IPv6 NAT support is available since kernel 3.7.

mangle:

This table is used for specialized packet alteration. Until kernel 2.4.17 it had two built-in chains: **PREROUTING** (for altering incoming packets before routing) and **OUTPUT** (for altering locally-generated packets before routing). Since kernel 2.4.18, three other built-in chains are also supported: **INPUT** (for packets coming into the box itself), **FORWARD** (for altering packets being routed through the box), and **POSTROUTING** (for altering packets as they are about to go out).

raw: This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target. It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables. It provides the following built-in chains: **PREROUTING** (for packets arriving via any network interface) **OUTPUT** (for packets generated by local processes)

security:

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the **SECMARK** and **CONNSECMARK** targets. Mandatory Access Control is implemented by Linux Security Modules such as SELinux. The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules. This table provides the following built-in chains: **INPUT** (for packets coming into the box itself), **OUTPUT** (for altering locally-generated packets before routing), and **FORWARD** (for altering packets being routed through the box).

OPTIONS

The options that are recognized by **iptables** and **ip6tables** can be divided into several different groups.

COMMANDS

These options specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below. For long versions of the command and option names, you need to use only enough letters to ensure that **iptables** can differentiate it from all other options.

-A, --append *chain rule-specification*

Append one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.

-C, --check *chain rule-specification*

Check whether a rule matching the specification does exist in the selected chain. This command uses the same logic as **-D** to find a matching entry, but does not alter the existing iptables configuration and uses its exit code to indicate success or failure.

-D, --delete *chain rule-specification*

-D, --delete *chain rulenum*

Delete one or more rules from the selected chain. There are two versions of this command: the rule can be specified as a number in the chain (starting at 1 for the first rule) or a rule to match.

-I, --insert *chain [rulenum] rule-specification*

Insert one or more rules in the selected chain as the given rule number. So, if the rule number is 1, the rule or rules are inserted at the head of the chain. This is also the default if no rule number is specified.

-R, --replace *chain rulenum rule-specification*

Replace a rule in the selected chain. If the source and/or destination names resolve to multiple addresses, the command will fail. Rules are numbered starting at 1.

-L, --list [chain]

List all rules in the selected chain. If no chain is selected, all chains are listed. Like every other iptables command, it applies to the specified table (filter is the default), so NAT rules get listed by `iptables -t nat -n -L`

Please note that it is often used with the **-n** option, in order to avoid long reverse DNS lookups. It is legal to specify the **-Z** (zero) option as well, in which case the chain(s) will be atomically listed and zeroed. The exact output is affected by the other arguments given. The exact rules are suppressed until you use

`iptables -L -v`
or `iptables-save(8)`.

-S, --list-rules [chain]

Print all rules in the selected chain. If no chain is selected, all chains are printed like `iptables-save`. Like every other iptables command, it applies to the specified table (filter is the default).

-F, --flush [chain]

Flush the selected chain (all the chains in the table if none is given). This is equivalent to deleting all the rules one by one.

-Z, --zero [chain [rulenum]]

Zero the packet and byte counters in all chains, or only the given chain, or only the given rule in a chain. It is legal to specify the **-L, --list** (list) option as well, to see the counters immediately before they are cleared. (See above.)

-N, --new-chain chain

Create a new user-defined chain by the given name. There must be no target of that name already.

-X, --delete-chain [chain]

Delete the optional user-defined chain specified. There must be no references to the chain. If there are, you must delete or replace the referring rules before the chain can be deleted. The chain must be empty, i.e. not contain any rules. If no argument is given, it will attempt to delete every non-built-in chain in the table.

-P, --policy chain target

Set the policy for the built-in (non-user-defined) chain to the given target. The policy target must be either **ACCEPT** or **DROP**.

-E, --rename-chain old-chain new-chain

Rename the user specified chain to the user supplied name. This is cosmetic, and has no effect on the structure of the table.

-h Help. Give a (currently very brief) description of the command syntax.**PARAMETERS**

The following parameters make up a rule specification (as used in the add, delete, insert, replace and append commands).

-4, --ipv4

This option has no effect in `iptables` and `iptables-restore`. If a rule using the **-4** option is inserted with (and only with) `ip6tables-restore`, it will be silently ignored. Any other uses will throw an error. This option allows IPv4 and IPv6 rules in a single rule file for use with both `iptables-restore` and `ip6tables-restore`.

-6, --ipv6

If a rule using the **-6** option is inserted with (and only with) `iptables-restore`, it will be silently ignored. Any other uses will throw an error. This option allows IPv4 and IPv6 rules in a single rule file for use with both `iptables-restore` and `ip6tables-restore`. This option has no effect in `ip6tables` and `ip6tables-restore`.

[!] -p, --protocol protocol

The protocol of the rule or of the packet to check. The specified protocol can be one of **tcp**, **udp**, **udplite**, **icmp**, **icmpv6**, **esp**, **ah**, **sctp**, **mh** or the special keyword "**all**", or it can be a numeric

value, representing one of these protocols or a different one. A protocol name from /etc/protocols is also allowed. A "!" argument before the protocol inverts the test. The number zero is equivalent to **all**. "all" will match with all protocols and is taken as default when this option is omitted. Note that, in ip6tables, IPv6 extension headers except **esp** are not allowed. **esp** and **ipv6-nonext** can be used with Kernel version 2.6.11 or later. The number zero is equivalent to **all**, which means that you cannot test the protocol field for the value 0 directly. To match on a HBH header, even if it were the last, you cannot use **-p 0**, but always need **-m hbh**.

[!] **-s, --source** *address[/mask][,...]*

Source specification. *Address* can be either a network name, a hostname, a network IP address (with */mask*), or a plain IP address. Hostnames will be resolved once only, before the rule is submitted to the kernel. Please note that specifying any name to be resolved with a remote query such as DNS is a really bad idea. The *mask* can be either an ipv4 network mask (for iptables) or a plain number, specifying the number of 1's at the left side of the network mask. Thus, an iptables mask of 24 is equivalent to 255.255.255.0. A "!" argument before the address specification inverts the sense of the address. The flag **--src** is an alias for this option. Multiple addresses can be specified, but this will **expand to multiple rules** (when adding with **-A**), or will cause multiple rules to be deleted (with **-D**).

[!] **-d, --destination** *address[/mask][,...]*

Destination specification. See the description of the **-s** (source) flag for a detailed description of the syntax. The flag **--dst** is an alias for this option.

-m, --match *match*

Specifies a match to use, that is, an extension module that tests for a specific property. The set of matches make up the condition under which a target is invoked. Matches are evaluated first to last as specified on the command line and work in short-circuit fashion, i.e. if one extension yields false, evaluation will stop.

-j, --jump *target*

This specifies the target of the rule; i.e., what to do if the packet matches it. The target can be a user-defined chain (other than the one this rule is in), one of the special builtin targets which decide the fate of the packet immediately, or an extension (see **EXTENSIONS** below). If this option is omitted in a rule (and **-g** is not used), then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

-g, --goto *chain*

This specifies that the processing should continue in a user specified chain. Unlike the **--jump** option return will not continue processing in this chain but instead in the chain that called us via **--jump**.

[!] **-i, --in-interface** *name*

Name of an interface via which a packet was received (only for packets entering the **INPUT**, **FORWARD** and **PREROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[!] **-o, --out-interface** *name*

Name of an interface via which a packet is going to be sent (for packets entering the **FORWARD**, **OUTPUT** and **POSTROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[!] **-f, --fragment**

This means that the rule only refers to second and further IPv4 fragments of fragmented packets. Since there is no way to tell the source or destination ports of such a packet (or ICMP type), such a packet will not match any rules which specify them. When the "!" argument precedes the **-f** flag, the rule will only match head fragments, or unfragmented packets. This option is IPv4 specific, it is not available in ip6tables.

-c, --set-counters *packets bytes*

This enables the administrator to initialize the packet and byte counters of a rule (during **INSERT**, **APPEND**, **REPLACE** operations).

OTHER OPTIONS

The following additional options can be specified:

-v, --verbose

Verbose output. This option makes the list command show the interface name, the rule options (if any), and the TOS masks. The packet and byte counters are also listed, with the suffix 'K', 'M' or 'G' for 1000, 1,000,000 and 1,000,000,000 multipliers respectively (but see the **-x** flag to change this). For appending, insertion, deletion and replacement, this causes detailed information on the rule or rules to be printed. **-v** may be specified multiple times to possibly emit more detailed debug statements.

-w, --wait [*seconds*]

Wait for the xtables lock. To prevent multiple instances of the program from running concurrently, an attempt will be made to obtain an exclusive lock at launch. By default, the program will exit if the lock cannot be obtained. This option will make the program wait (indefinitely or for optional *seconds*) until the exclusive lock can be obtained.

-W, --wait-interval *microseconds*

Interval to wait per each iteration. When running latency sensitive applications, waiting for the xtables lock for extended durations may not be acceptable. This option will make each iteration take the amount of time specified. The default interval is 1 second. This option only works with **-w**.

-n, --numeric

Numeric output. IP addresses and port numbers will be printed in numeric format. By default, the program will try to display them as host names, network names, or services (whenever applicable).

-x, --exact

Expand numbers. Display the exact value of the packet and byte counters, instead of only the rounded number in K's (multiples of 1000) M's (multiples of 1000K) or G's (multiples of 1000M). This option is only relevant for the **-L** command.

--line-numbers

When listing rules, add line numbers to the beginning of each rule, corresponding to that rule's position in the chain.

--modprobe=*command*

When adding or inserting rules into a chain, use *command* to load any necessary modules (targets, match extensions, etc).

LOCK FILE

iptables uses the */run/xtables.lock* file to take an exclusive lock at launch.

The **XTABLES_LOCKFILE** environment variable can be used to override the default setting.

MATCH AND TARGET EXTENSIONS

iptables can use extended packet matching and target modules. A list of these is available in the **iptables-extensions(8)** manpage.

DIAGNOSTICS

Various error messages are printed to standard error. The exit code is 0 for correct functioning. Errors which appear to be caused by invalid or abused command line parameters cause an exit code of 2, and other errors cause an exit code of 1.

BUGS

Bugs? What's this? ;-) Well, you might want to have a look at <http://bugzilla.netfilter.org/>

COMPATIBILITY WITH IPCHAINS

This **iptables** is very similar to ipchains by Rusty Russell. The main difference is that the chains **INPUT** and **OUTPUT** are only traversed for packets coming into the local host and originating from the local host respectively. Hence every packet only passes through one of the three chains (except loopback traffic, which involves both INPUT and OUTPUT chains); previously a forwarded packet would pass through all three.

The other main difference is that **-i** refers to the input interface; **-o** refers to the output interface, and both are available for packets entering the **FORWARD** chain.

The various forms of NAT have been separated out; **iptables** is a pure packet filter when using the default 'filter' table, with optional extension modules. This should simplify much of the previous confusion over the combination of IP masquerading and packet filtering seen previously. So the following options are handled differently:

-j MASQ
-M -S
-M -L

There are several other changes in iptables.

SEE ALSO

iptables-apply(8), **iptables-save(8)**, **iptables-restore(8)**, **iptables-extensions(8)**,

The packet-filtering-HOWTO details iptables usage for packet filtering, the NAT-HOWTO details NAT, the netfilter-extensions-HOWTO details the extensions that are not in the standard distribution, and the netfilter-hacking-HOWTO details the netfilter internals.

See <http://www.netfilter.org/>.

AUTHORS

Rusty Russell originally wrote iptables, in early consultation with Michael Neuling.

Marc Boucher made Rusty abandon ipnatctl by lobbying for a generic packet selection framework in iptables, then wrote the mangle table, the owner match, the mark stuff, and ran around doing cool stuff everywhere.

James Morris wrote the TOS target, and tos match.

Jozsef Kadlecik wrote the REJECT target.

Harald Welte wrote the ULOG and NFQUEUE target, the new libiptc, as well as the TTL, DSCP, ECN matches and targets.

The Netfilter Core Team is: Jozsef Kadlecik, Pablo Neira Ayuso, Eric Leblond, Florian Westphal and Arturo Borrero Gonzalez. Emeritus Core Team members are: Marc Boucher, Martin Josefsson, Yasuyuki Kozakai, James Morris, Harald Welte and Rusty Russell.

Man page originally written by Herve Eychenne <rv@wallfire.org>.

VERSION

This manual page applies to iptables/ip6tables 1.8.7.

ipsec(8) - Linux man page

Name

ipsec - invoke IPsec utilities

Synopsis

ipsec *command* [*argument...*] **ipsec** --help

ipsec --version
ipsec --versioncode
ipsec --copyright
ipsec --directory
ipsec --confdir

Description

Ipsec invokes any of several utilities involved in controlling the IPsec encryption/authentication system, running the specified *command* with the specified *arguments* as if it had been invoked directly. This largely eliminates possible name collisions with other software, and also permits some centralized services.

In particular, **ipsec** supplies the invoked *command* with a suitable PATH environment variable, and also provides IPSEC_DIR, IPSEC_CONFS, and IPSEC_VERSION environment variables, containing respectively the full pathname of the directory where the IPsec utilities are stored, the full pathname of the directory where the configuration files live, and the IPsec version number.

ipsec --help lists the available commands. Most have their own manual pages, e.g. **ipsec auto(8)** for *auto*.

ipsec --version outputs version information about Linux FreeS/WAN. A version code of the form "Uxxx/Kyyy" indicates that the user-level utilities are version xxx but the kernel portion appears to be version yyy (this form is used only if the two disagree).

ipsec --versioncode outputs *just* the version code, with none of **--version**'s supporting information, for use by scripts.

ipsec --copyright supplies boring copyright details.

ipsec --directory reports where **ipsec** thinks the IPsec utilities are stored.

ipsec --confdir reports where **ipsec** thinks the IPsec configuration files are stored.

Files

/usr/local/lib/ipsec usual utilities directory

Environment

The following environment variables control where FreeS/WAN finds its components. The **ipsec** command sets them if they are not already set.

IPSEC_EXECDIR

directory containing published commands

IPSEC_LIBDIR

directory containing internal executables

IPSEC_SBINDIR

directory containing **ipsec** command

IPSEC_CONFS

directory containing configuration files

See Also

ipsec.**conf**(5), ipsec.**secrets**(5), **ipsec_auto**(8), **ipsec_barf**(8), **ipsec_setup**(8),
ipsec_showdefaults(8), **ipsec_showhostkey**(8)

HTML documentation shipped with the release, starting with *doc/index.html*.

<<http://www.freeswan.org/doc.html>> may also be of use.

History

Written for Linux FreeS/WAN <<http://www.freeswan.org>> by Henry Spencer.

Bugs

The provision of centralized services, while convenient, does compromise the original concept of making the utilities invocable directly as well as via **ipsec**.

Referenced By

ipsec.conf(5), **ipsec.secrets**(5), **ipsec_confread**(8), **ipsec_copyright**(8),
ipsec_include(8), **ipsec_keycensor**(8), **ipsec_plutoload**(8), **ipsec_plutorun**(8),
ipsec_realsetup(8), **ipsec_secretcensor**(8), **ipsec_startklips**(8),
ipsec_updown(8), **ipsec_updown.bsdkame**(8), **ipsec_updown.netkey**(8),
ipsec_eroute(5), **ipsec_eroute**(8), **ipsec_ipsec.conf**(5), **ipsec_ipsec_pluto**(8),
ipsec_ipsec_spi(5), **ipsec_ipsec_spi**(8), **ipsec_klipsdebug**(5), **ipsec_klipsdebug**(8),

[ipsec_look\(8\)](#), **[ipsec_pf_key\(5\)](#)**, **[ipsec_pf_key\(8\)](#)**, **[ipsec_pluto\(8\)](#)**, **[ipsec_policy\(8\)](#)**,
[ipsec_selinux\(8\)](#), **[ipsec_showpolicy\(8\)](#)**, **[ipsec_spi\(5\)](#)**, **[ipsec_spi\(8\)](#)**, **[ipsec_spigrp\(5\)](#)**,
[ipsec_spigrp\(8\)](#), **[ipsec_tncfg\(5\)](#)**, **[ipsec_tncfg\(8\)](#)**, **[ipsec_trap_count\(5\)](#)**,
[ipsec_trap_sendcount\(5\)](#), **[ipsec_version\(5\)](#)**, **[strongswan.conf\(5\)](#)**,
[strongswan_updown\(8\)](#), **[strongswan_updown_espmark\(8\)](#)**,
[strongswan_ipsec.conf\(5\)](#), **[strongswan_ipsec.secrets\(5\)](#)**, **[strongswan_pluto\(8\)](#)**

ipsec.conf(5) - Linux man page

Name

ipsec.conf - IPsec configuration and connections

Description

The optional *ipsec.conf* file specifies most configuration and control information for the Openswan IPsec subsystem. (The major exception is secrets for authentication; see [ipsec.secrets\(5\)](#).) Its contents are not security-sensitive *unless* manual keying is being done for more than just testing, in which case the encryption/authentication keys in the descriptions for the manually-keyed connections are very sensitive (and those connection descriptions are probably best kept in a separate file, via the include facility described below).

The file is a text file, consisting of one or more *sections*. White space followed by **#** followed by anything to the end of the line is a comment and is ignored, as are empty lines which are not within a section.

A line which contains **include** and a file name, separated by white space, is replaced by the contents of that file, preceded and followed by empty lines. If the file name is not a full pathname, it is considered to be relative to the directory containing the including file. Such inclusions can be nested. Only a single filename may be supplied, and it may not contain white space, but it may include shell wildcards (see [sh\(1\)](#)); for example:

include ipsec.*.conf

The intention of the include facility is mostly to permit keeping information on connections, or sets of connections, separate from the main configuration file. This permits such connection descriptions to be changed, copied to the other security gateways involved, etc., without having to constantly extract them from the configuration file and then insert them back into it. Note also the **also** and **alsoflip** parameters (described below) which permit splitting a single logical section (e.g. a connection description) into several actual sections.

The first significant line of the file must specify the version of this specification that it conforms to:

version 2

A section begins with a line of the form:

type name

where *type* indicates what type of section follows, and *name* is an arbitrary name which distinguishes the section from others of the same type. (Names must start with a letter and may contain only letters, digits, periods, underscores, and hyphens.) All subsequent non-empty lines which begin with white space are part of the section; comments within a section must begin with white space too. There may be only one section of a given type with a given name.

Lines within the section are generally of the form

parameter=value

(note the mandatory preceding white space). There can be white space on either side of the **=**. Parameter names follow the same syntax as section names, and are specific to a section type. Unless otherwise explicitly specified, no parameter name may appear more than once in a section.

An empty *value* stands for the system default value (if any) of the parameter, i.e. it is roughly equivalent to omitting the parameter line entirely. A *value* may contain white space only if the entire *value* is enclosed in double quotes (""); a *value* cannot itself contain a double quote, nor may it be continued across more than one line.

Numeric values are specified to be either an "integer" (a sequence of digits) or a "decimal number" (sequence of digits optionally followed by '.' and another sequence of digits).

There is currently one parameter which is available in any type of section:

also

the value is a section name; the parameters of that section are appended to this section, as if they had been written as part of it. The specified section must exist, must follow the current one, and must have the same section type. (Nesting is permitted, and there may be more than one **also** in a single section, although it is forbidden to append the same section more than once.) This allows, for example, keeping the encryption keys for a connection in a separate file from the rest of the description, by using both an **also** parameter and an **include** line. (Caution, see BUGS below for some restrictions.)

alsoflip

can be used in a **conn** section. It acts like an **also** that flips the referenced section's entries left-for-right.

Parameter names beginning with **x-** (or **X-**, or **x_**, or **X_**) are reserved for user extensions and will never be assigned meanings by IPsec. Parameters with such names must still observe the syntax rules (limits on characters used in the name; no white space in a non-quoted value; no newlines or double quotes within the value). All other as-yet-unused parameter names are reserved for future IPsec improvements.

A section with name **%default** specifies defaults for sections of the same type. For each parameter in it, any section of that type which does not have a parameter of the same name gets a copy of the one from the **%default** section. There may be multiple **%default** sections of a given type, but only one default may be supplied for any specific parameter name, and all **%default** sections of a given type must precede all non-**%default** sections of that type. **%default** sections may not contain **also** or **alsoflip** parameters.

Currently there are two types of section: a **config** section specifies general configuration information for IPsec, while a **conn** section specifies an IPsec connection.

Conn Sections

A **conn** section contains a *connection specification*, defining a network connection to be made using IPsec. The name given is arbitrary, and is used to identify the connection to **ipsec_auto**(8) and **ipsec_manual**(8). Here's a simple example:

```
conn snt
left=10.11.11.1
```

```
leftsubnet=10.0.1.0/24
leftnexthop=172.16.55.66
leftsourceip=10.0.1.1
right=192.168.22.1
rightsubnet=10.0.2.0/24
rightnexthop=172.16.88.99
rightsourceip=10.0.2.1
```

keyingtries=%forever

A note on terminology... In automatic keying, there are two kinds of communications going on: transmission of user IP packets, and gateway-to-gateway negotiations for keying, rekeying, and general control. The data path (a set of "IPsec SAs") used for user packets is herein referred to as the "connection"; the path used for negotiations (built with "ISAKMP SAs") is referred to as the "keying channel".

To avoid trivial editing of the configuration file to suit it to each system involved in a connection, connection specifications are written in terms of *left* and *right* participants, rather than in terms of local and remote. Which participant is considered *left* or *right* is arbitrary; IPsec figures out which one it is being run on based on internal information. This permits using identical connection specifications on both ends. There are cases where there is no symmetry; a good convention is to use *left* for the local side and *right* for the remote side (the first letters are a good mnemonic).

Many of the parameters relate to one participant or the other; only the ones for *left* are listed here, but every parameter whose name begins with **left** has a **right** counterpart, whose description is the same but with **left** and **right** reversed.

Parameters are optional unless marked "(required)"; a parameter required for manual keying need not be included for a connection which will use only automatic keying, and vice versa.

CONN PARAMETERS: GENERAL

The following parameters are relevant to both automatic and manual keying. Unless otherwise noted, for a connection to work, in general it is necessary for the two ends to agree exactly on the values of these parameters.

connaddrfamily

the connection address family of the connection; currently the accepted values are **ipv4** (the default); or **ipv6**,

The ipv6 family is currently only supported using the NETKEY stack.

type

the type of the connection; currently the accepted values are **tunnel** (the default) signifying a host-to-host, host-to-subnet, or subnet-to-subnet tunnel; **transport**, signifying host-to-host transport mode; **passthrough**, signifying that no IPsec processing should be done at all; **drop**,

signifying that packets should be discarded; and **reject**, signifying that packets should be discarded and a diagnostic ICMP returned.

left

(required) the IP address of the left participant's public-network interface, in any form accepted by **ipsec_ttoaddr**(3). Currently, IPv4 and IPv6 IP addresses are supported. There are several magic values. If it is **%defaultroute**, and the **config setup** section's, **interfaces** specification contains **%defaultroute**, **left** will be filled in automatically with the local address of the default-route interface (as determined at IPsec startup time); this also overrides any value supplied for **leftnexthop**. (Either **left** or **right** may be **%defaultroute**, but not both.) The value **%any** signifies an address to be filled in (by automatic keying) during negotiation. The value **%opportunistic** signifies that both **left** and **leftnexthop** are to be filled in (by automatic keying) from DNS data for **left**'s client. The value can also contain the interface name, which will then later be used to obtain the IP address from to fill in. For example **%ppp0** The values **%group** and **%opportunisticgroup** makes this a policy group conn: one that will be instantiated into a regular or opportunistic conn for each CIDR block listed in the policy group file with the same name as the conn.

If using IP addresses in combination with NAT, always use the actual local machine's (NAT'ed) IP address, and if the remote (eg **right=**) is NAT'ed as well, the remote's public (**not** NAT'ed) IP address. Note that this makes the configuration no longer symmetrical on both sides, so you cannot use an identical configuration file on both hosts.

leftsubnet

private subnet behind the left participant, expressed as *network/netmask* (actually, any form acceptable to **ipsec_ttosubnet**(3)); Currently, IPv4 and IPv6 ranges are supported. If omitted, essentially assumed to be */left/32*, signifying that the left end of the connection goes to the left participant only

leftsubnets

specify multiple private subnets behind the left participant, expressed as { *networkA/netmaskA* *networkB/netmaskB* [...] } If both a **leftsubnets=** and **rightsubnets=** is defined, all combinations of subnet tunnels will be instantiated. You cannot use **leftsubnet** and **leftsubnets** together. For examples see *testing/pluto/multinet-**.

leftprotoport

allowed protocols and ports over connection, also called Port Selectors. The argument is in the form *protocol*, which can be a number or a name that will be looked up in */etc/protocols*, such as *leftprotoport=icmp*, or in the form of *protocol/port*, such as *tcp/smtp*. Ports can be defined as a number (eg. 25) or as a name (eg *smtp*) which will be looked up in */etc/services*. A special keyword **%any** can be used to allow all ports of a certain protocol. The most common use of this option is for L2TP connections to only allow I2tp packets (UDP port 1701), eg:

leftprotoport=17/1701. Some clients, notably older Windows XP and some Mac OSX clients, use a random high port as source port. In those cases *rightprotoport=17/%any* can be used to allow all UDP traffic on the connection. Note that this option is part of the proposal, so it cannot be arbitrarily left out if one end does not care about the traffic selection over this connection - both peers have to agree. The Port Selectors show up in the output of *ipsec eroute* and *ipsec auto --status* eg: "*I2tp*": 193.110.157.131[@aivd.xelernace.com]:7/1701...%any:17/1701 This option only filters outbound traffic. Inbound traffic selection must still be based on firewall rules activated by an *updown* script. The variables \$PLUTO_MY_PROTOCOL, \$PLUTO_PEER_PROTOCOL, \$PLUTO_MY_PORT, and \$PLUTO_PEER_PORT are available for use in *updown* scripts. Older workarounds for bugs involved a setting of 17/0 to denote *any single UDP*

port (not UDP port 0). Some clients, most notably OSX, uses a random high port, instead of port 1705 for L2TP.

leftnexthop

next-hop gateway IP address for the left participant's connection to the public network; defaults to **%direct** (meaning *right*). If the value is to be overridden by the **left=%defaultroute** method (see above), an explicit value must *not* be given. If that method is not being used, but **leftnexthop** is **%defaultroute**, and **interfaces=%defaultroute** is used in the **config setup** section, the next-hop gateway address of the default-route interface will be used. The magic value **%direct** signifies a value to be filled in (by automatic keying) with the peer's address. Relevant only locally, other end need not agree on it.

leftsourceip

the IP address for this host to use when transmitting a packet to the other side of this link. Relevant only locally, the other end need not agree. This option is used to make the gateway itself use its internal IP, which is part of the **leftsubnet**, to communicate to the **rightsubnet** or **right**. Otherwise, it will use its **nearest** IP address, which is its public IP address. This option is mostly used when defining subnet-subnet connections, so that the gateways can talk to each other and the subnet at the other end, without the need to build additional host-subnet, subnet-host and host-host tunnels. Both IPv4 and IPv6 addresses are supported.

leftupdown

what "updown" script to run to adjust routing and/or firewalling when the status of the connection changes (default **ipsec _updown**). May include positional parameters separated by white space (although this requires enclosing the whole string in quotes); including shell metacharacters is unwise. An example to enable routing when using the NETKEY stack, one can use:

```
leftupdown="ipsec _updown --route yes"
```

See [**ipsec_pluto**\(8\)](#) for details. Relevant only locally, other end need not agree on it.

leftfirewall

This option is obsolete and should not be used anymore.

If one or both security gateways are doing forwarding firewalling (possibly including masquerading), and this is specified using the **firewall** parameters, tunnels established with IPsec are exempted from it so that packets can flow unchanged through the tunnels. (This means that all subnets connected in this manner must have distinct, non-overlapping subnet address blocks.) This is done by the default **updown** script (see [**ipsec_pluto**\(8\)](#)).

The implementation of this makes certain assumptions about firewall setup, and the availability of the *Linux Advanced Routing* tools. In situations calling for more control, it may be preferable for the user to supply his own **updown** script, which makes the appropriate adjustments for his system.

CONN PARAMETERS: AUTOMATIC KEYING

The following parameters are relevant only to automatic keying, and are ignored in manual keying. Unless otherwise noted, for a connection to work, in general it is necessary for the two ends to agree exactly on the values of these parameters.

auto

what operation, if any, should be done automatically at IPsec startup; currently-accepted values are **add** (signifying an **ipsec auto --add**), **route** (signifying that plus an **ipsec auto --route**),

start (signifying that plus an **ipsec auto --up**), **manual** (signifying an **ipsec manual --up**), and **ignore** (also the default) (signifying no automatic startup operation). See the **config setup** discussion below. Relevant only locally, other end need not agree on it (but in general, for an intended-to-be-permanent connection, both ends should use **auto=start** to ensure that any reboot causes immediate renegotiation).

authby

how the two security gateways should authenticate each other; acceptable values are **secret** for shared secrets, **rsasig** for RSA digital signatures (the default), **secret|rsasig** for either, and **never** if negotiation is never to be attempted or accepted (useful for shunt-only conns). Digital signatures are superior in every way to shared secrets.

ike

IKE encryption/authentication algorithm to be used for the connection (phase 1 aka ISAKMP SA). The format is "*cipher-hash;modpgroup, cipher-hash;modpgroup, ...*" Any left out option will be filled in with all allowed default options. Multiple proposals are separated by a comma. If an **ike=** line is specified, no other received proposals will be accepted. Formerly there was a distinction (by using a "!" symbol) between "strict mode" or not. That mode has been obsoleted. If an **ike=** option is specified, the mode is always strict, meaning no other received proposals will be accepted. Some examples are **ike=3des-sha1,aes-sha1, ike=aes, ike=aes128-md5;modp2048, ike=aes128-sha1;dh22, ike=3des-md5;modp1024,aes-sha1;modp1536** or **ike=modp1536**. The options must be suitable as a value of **ipsec_spi**(8)'s **--ike** option. The default is to use IKE, and to allow all combinations of:

cipher:	3des or aes
hash:	sha1 or md5
pfs group (DHgroup):	modp1024 or modp1536

If Openswan was compiled with extra INSECURE and BROKEN options, then the des (1des) and null cipher, as well as modp768 are available. This turns your VPN into a joke. Do not enable these options.

If openswan was compiled with USE_MODP_RFC5114 support, then Diffie-Hellman groups 22, 23 and 24 are also implemented as per RFC-5114. Instead of the modp key syntax, use the "dh" keyword, for example *ike=3des-sha1;dh23*

phase2

Sets the type of SA that will be produced. Valid options are: **esp** for encryption (the default), and **ah** for authentication only.

phase2alg

Specifies the algorithms that will be offered/accepted for a phase2 negotiation. If not specified, a secure set of defaults will be used. Sets are separated using comma's.

The default values are the same as for **ike=**. Note also that not all ciphers available to the kernel (eg through CryptoAPI) are necessarily supported here.

The format for ESP is ENC-AUTH followed by an optional PFSgroup. For instance, "3des-md5" or "aes256-sha1;modp2048" or "aes-sha1,aes-md5".

For RFC-5114 DH groups, use the "dh" keyword, eg "aes256-sha1;dh23"

The format for AH is AUTH followed by an optional PFSgroup. For instance, "md5" or "sha1;modp1536".

A special case is AES CCM, which uses the syntax of "phase2alg=aes_ccm_a-152-null"

sha2_truncbug

The default hash truncation for sha2_256 is 128 bits. Linux implemented the draft version which stated 96 bits. This option enables using the bad 96 bits version to interop with older linux kernels (unpatched version 2.6.33 and older) and openswan versions before 2.6.38. Currently the accepted values are **no**, (the default) signifying default IETF truncation of 128 bits, or **yes**, signifying 96 bits broken Linux kernel style truncation.

esp

This option is obsolete. Please use **phase2alg** instead.

ah

AH authentication algorithm to be used for the connection, e.g here. **hmac-md5** The options must be suitable as a value of **ipsec_spi**(8)'s **--ah** option. The default is not to use AH. If for some (invalid) reason you still think you need AH, please use esp with the null encryption cipher instead. Note also that not all ciphers available to the kernel (eg through CryptoAPI) are necessarily supported here.

ikev2

IKEv2 (RFC4309) settings to be used. Currently the accepted values are **permit**, (the default) signifying no IKEv2 should be transmitted, but will be accepted if the other ends initiates to us with IKEv2; **never** or **no** signifying no IKEv2 negotiation should be transmitted or accepted; **propose** or **yes** signifying that we permit IKEv2, and also use it as the default to initiate; **insist**, signifying we only accept and receive IKEv2 - IKEv1 negotiations will be rejected.

If the ikev2= setting is set to **permit** or **propose**, Openswan will try and detect a "bid down" attack from IKEv2 to IKEv1. Since there is no standard for transmitting the IKEv2 capability with IKEv1, Openswan uses a special Vendor ID "CAN-IKEv2". If a fall back from IKEv2 to IKEv1 was detected, and the IKEv1 negotiation contains Vendor ID "CAN-IKEv2", Openswan will immediately attempt and IKEv2 rekey and refuse to use the IKEv1 connection. With an ikev2= setting of **insist**, no IKEv1 negotiation is allowed, and no bid down attack is possible.

sareftrack

Set the method of tracking reply packets with SArefs when using an SAref compatible stack. Currently only the *mas* stack supports this. Acceptable values are **yes** (the default), **no** or **conntrack**. This option is ignored when SArefs are not supported. This option is passed as PLUTO_SAREF_TRACKING to the *updown* script which makes the actual decisions whether to perform any iptables/ip_conntrack manipulation. A value of yes means that an IPSEC mangle table will be created. This table will be used to match reply packets. A value of conntrack means that additionally, subsequent packets using this connection will be marked as well, reducing the lookups needed to find the proper SAref by using the ip_conntrack state. A value of no means no IPSEC mangle table is created, and SAref tracking is left to a third-party (kernel) module. In case of a third party module, the SArefs can be relayed using the HAVE_STATSD deamon.

leftid

how the left participant should be identified for authentication; defaults to **left**. Can be an IP address (in any **ipsec_ttoaddr**(3) syntax) or a fully-qualified domain name preceded by @ (which is used as a literal string and not resolved). The magic value **%fromcert** causes the ID to be set to a DN taken from a certificate that is loaded. Prior to 2.5.16, this was the default if a certificate was specified. The magic value **%onone** sets the ID to no ID. This is included for completeness, as the ID may have been set in the default conn, and one wishes for it to default instead of being explicitly set. The magic value **%myid** stands for the current setting of *myid*. This is set in **config setup** or by **ipsec_whack**(8)), or, if not set, it is the IP address in

%defaultroute (if that is supported by a TXT record in its reverse domain), or otherwise it is the system's hostname (if that is supported by a TXT record in its forward domain), or otherwise it is undefined.

leftrsasigkey

the left participant's public key for RSA signature authentication, in RFC 2537 format using **ipsec_todata**(3) encoding. The magic value **%none** means the same as not specifying a value (useful to override a default). The value **%dnsondemand** (the default) means the key is to be fetched from DNS at the time it is needed. The value **%dnsload** means the key is to be fetched from DNS at the time the connection description is read from *ipsec.conf*; currently this will be treated as **%none** if **right=%any** or **right=%opportunistic**. The value **%dns** is currently treated as **%dnsload** but will change to **%dnsondemand** in the future. The identity used for the left participant must be a specific host, not **%any** or another magic value. The value **%cert** will load the information required from a certificate defined in **%leftcert** and automatically define **leftid** for you. **Caution:** if two connection descriptions specify different public keys for the same **leftid**, confusion and madness will ensue.

leftrsasigkey2

if present, a second public key. Either key can authenticate the signature, allowing for key rollover.

leftcert

If you are using **leftrsasigkey=%cert** this defines the certificate you would like to use. It should point to a X.509 encoded certificate file. If you do not specify a full pathname, by default it will look in /etc/ipsec.d/certs. If openswan has been compiled with **USE_LIBNSS=true**, then openswan will also check the NSS database for RSA keys. These can be software or hardware.

leftca

specifies the authorized Certificate Authority (CA) that signed the certificate of the peer. If undefined, it defaults to the CA that signed the certificate specified in **leftcert**. The special **rightca=%same** is implied when not specifying a **rightca** and means that only peers with certificates signed by the same CA as the **leftca** will be allowed. This option is only useful in complex multi CA certificate situations. When using a single CA, it can be safely omitted for both left and right.

leftsendcert

This option configures when Openswan will send X.509 certificates to the remote host. Acceptable values are **yes|always** (signifying that we should always send a certificate), **ifasked** (signifying that we should send a certificate if the remote end asks for it), and **no|never** (signifying that we will never send a X.509 certificate). The default for this option is **ifasked** which may break compatibility with other vendor's IPSec implementations, such as Cisco and SafeNet. If you find that you are getting errors about no ID/Key found, you likely need to set this to **always**. This per-conn option replaces the obsolete global **nocrsend** option.

leftxauthserver

Left is an XAUTH server. This can use PAM for authentication or md5 passwords in /etc/ipsec.d/passwd. These are additional credentials to verify the user identity, and should not be confused with the XAUTH **group secret**, which is just a regular PSK defined in *ipsec.secrets*. The other side of the connection should be configured as **rightxauthclient**. XAUTH connections cannot rekey, so **rekey=no** should be specified in this conn. For further details on how to compile and use XAUTH, see README.XAUTH. Acceptable values are **yes** or **no** (the default).

leftxauthclient

Left is an XAUTH client. The xauth connection will have to be started interactively and cannot be configured using **auto=start**. Instead, it has to be started from the commandline using *ipsec auto --up connname*. You will then be prompted for the username and password. To setup an

XAUTH connection non-interactively, which defeats the whole purpose of XAUTH, but is regularly requested by users, it is possible to use a whack command - *ipsec whack --name baduser --ipsecgroup-xauth --xauthname badusername --xauthpass password --initiate* The other side of the connection should be configured as **rightxauthserver**. Acceptable values are **yes** or **no** (the default).

leftxauthusername

The XAUTH username associated with this XAUTH connection. The XAUTH password can be configured in the *ipsec.secrets* file.

leftmodecfgserver

Left is a Mode Config server. It can push network configuration to the client. Acceptable values are **yes** or **no** (the default).

leftmodecfgclient

Left is a Mode Config client. It can receive network configuration from the server. Acceptable values are **yes** or **no** (the default).

modecfgpull

Pull the Mode Config network information from the server. Acceptable values are **yes** or **no** (the default).

modecfgdns1, modecfgdns2, modecfgwins1, modecfgwins2

Specify the IP address for DNS or WINS servers for the client to use.

remote_peer_type

Set the remote peer type. This can enable additional processing during the IKE negotiation. Acceptable values are **cisco** or **ietf** (the default). When set to cisco, support for Cisco IPsec gateway redirection and Cisco obtained DNS and domainname are enabled. This includes automatically updating (and restoring) /etc/resolv.conf. These options require that XAUTH is also enabled on this connection.

nm_configured

Mark this connection as controlled by Network Manager. Acceptable values are **yes** or **no** (the default). Currently, setting this to yes will cause openswan to skip reconfiguring resolv.conf when used with XAUTH and ModeConfig.

forceencaps

In some cases, for example when ESP packets are filtered or when a broken IPsec peer does not properly recognise NAT, it can be useful to force RFC-3948 encapsulation. **forceencaps=yes** forces the NAT detection code to lie and tell the remote peer that RFC-3948 encapsulation (ESP in UDP port 4500 packets) is required. For this option to have any effect, the setup section option **nat_traversal=yes** needs to be set. Acceptable values are **yes** or **no** (the default).

overlapip

a boolean (yes/no) that determines, when *subnet=vhos: is used, if the virtual IP claimed by this states created from this connection can with states created from other connections.

Note that connection instances created by the Opportunistic Encryption or PKIX (x.509) instantiation system are distinct internally. They will inherit this policy bit.

The default is no.

This feature is only available with kernel drivers that support SAs to overlapping conns. At present only the (klips)mast protocol stack supports this feature.

dpddelay

Set the delay (in seconds) between Dead Peer Detection (RFC 3706) keepalives (R_U_THERE, R_U_THERE_ACK) that are sent for this connection (default 30 seconds). If dpddelay is set,

`dpdtimeout` also needs to be set.

dpdtimeout

Set the length of time (in seconds) we will idle without hearing either an R_U_THERE poll from our peer, or an R_U_THERE_ACK reply. After this period has elapsed with no response and no traffic, we will declare the peer dead, and remove the SA (default 120 seconds). If `dpdtimeout` is set, `dpdaction` also needs to be set.

dpdaction

When a DPD enabled peer is declared dead, what action should be taken. **hold** (default) means the eroute will be put into %hold status, while **clear** means the eroute and SA with both be cleared. **restart** means the the SA will immediately be renegotiated, and **restart_by_peer** means that *ALL* SA's to the dead peer will renegotiated.

`dpdaction=clear` is really only useful on the server of a Road Warrior config.

pfs

whether Perfect Forward Secrecy of keys is desired on the connection's keying channel (with PFS, penetration of the key-exchange protocol does not compromise keys negotiated earlier); Since there is no reason to ever refuse PFS, Openswan will allow a connection defined with **pfs=no** to use PFS anyway. Acceptable values are **yes** (the default) and **no**.

pfsgroup

This option is obsoleted, please use `phase2alg` if you need the pfs to be different from `phase1` (the default) using: `phase2alg=aes128-md5;modp1024`

aggrmode

Use Aggressive Mode instead of Main Mode. Aggressive Mode is less secure, and vulnerable to Denial Of Service attacks. It is also vulnerable to brute force attacks with software such as **ikecrack**. It should not be used, and it should especially not be used with XAUTH and group secrets (PSK). If the remote system administrator insists on staying irresponsible, enable this option.

Aggressive Mode is further limited to only proposals with one DH group as there is no room to negotiate the DH group. Therefor it is mandatory for Aggressive Mode connections that both **ike=** and **phase2alg=** options are specified with only fully specified proposal using one DH group. Acceptable values are **yes** or **no** (the default).

The ISAKMP SA is created in exchange 1 in aggressive mode. Openswan has to send the exponent during that exchange, so it has to know what DH group to use before starting. This is why you can not have multiple DH groups in aggressive mode. In IKEv2, which uses a similar method to IKEv1 Aggressive Mode, there is a message to convey the DH group is wrong, and so an IKEv2 connection can actually recover from picking the wrong DH group by restarting its negotiation.

salifetime

how long a particular instance of a connection (a set of encryption/authentication keys for user packets) should last, from successful negotiation to expiry; acceptable values are an integer optionally followed by **s** (a time in seconds) or a decimal number followed by **m**, **h**, or **d** (a time in minutes, hours, or days respectively) (default **8h**, maximum **24h**). Normally, the connection is renegotiated (via the keying channel) before it expires. The two ends need not exactly agree on **salifetime**, although if they do not, there will be some clutter of superseded connections on the end which thinks the lifetime is longer.

The keywords "keylife" and "lifetime" are aliases for "salifetime."

rekey

whether a connection should be renegotiated when it is about to expire; acceptable values are **yes** (the default) and **no**. The two ends need not agree, but while a value of **no** prevents Pluto from requesting renegotiation, it does not prevent responding to renegotiation requested from the other end, so **no** will be largely ineffective unless both ends agree on it.

rekeymargin

how long before connection expiry or keying-channel expiry should attempts to negotiate a replacement begin; acceptable values as for **salifetime** (default **9m**). Relevant only locally, other end need not agree on it.

rekeyfuzz

maximum percentage by which **rekeymargin** should be randomly increased to randomize rekeying intervals (important for hosts with many connections); acceptable values are an integer, which may exceed 100, followed by a '%' (default set by [ipsec_pluto](#)(8), currently **100%**). The value of **rekeymargin**, after this random increase, must not exceed **salifetime**. The value **0%** will suppress time randomization. Relevant only locally, other end need not agree on it.

keyingtries

how many attempts (a whole number or **%forever**) should be made to negotiate a connection, or a replacement for one, before giving up (default **%forever**). The value **%forever** means "never give up" (obsolete: this can be written 0). Relevant only locally, other end need not agree on it.

ikelifetime

how long the keying channel of a connection (buzzphrase: "ISAKMP SA") should last before being renegotiated; acceptable values as for **keylife** (default set by [ipsec_pluto](#)(8), currently **1h**, maximum **24h**). The two-ends-disagree case is similar to that of **keylife**.

compress

whether IPComp compression of content is proposed on the connection (link-level compression does not work on encrypted data, so to be effective, compression must be done *before* encryption); acceptable values are **yes** and **no** (the default). The two ends need not agree. A value of **yes** causes IPsec to propose both compressed and uncompressed, and prefer compressed. A value of **no** prevents IPsec from proposing compression; a proposal to compress will still be accepted.

metric

Set the metric for the routes to the ipsecX or mastX interface. This makes it possible to do host failover from another interface to ipsec using route management. This value is passed to the _updown scripts as PLUTO_METRIC. This option is only available with KLIPS or MAST on Linux. Acceptable values are positive numbers, with the default being **1**.

disablearrivalcheck

whether KLIPS's normal tunnel-exit check (that a packet emerging from a tunnel has plausible addresses in its header) should be disabled; acceptable values are **yes** and **no** (the default). Tunnel-exit checks improve security and do not break any normal configuration. Relevant only locally, other end need not agree on it.

failureshunt

what to do with packets when negotiation fails. The default is **none**: no shunt; **passthrough**, **drop**, and **reject** have the obvious meanings.

CONN PARAMETERS: MANUAL KEYING

This command was obsoleted around the same time that Al Gore invented the internet. ipsec manual was used in the jurassic period to load static keys into the kernel. There are no rational reasons to use this, and it is not supported anymore. If you need to create static SAs, then you can use **ipsec spi** and **ipsec eroute** when using KLIPS or **ip xfrm** or **setkey** when using NETKEY.

No rational person uses static keys. They are not easier to use. REPEAT: they are not easier to use.

Config Sections

At present, the only **config** section known to the IPsec software is the one named **setup**, which contains information used when the software is being started (see [**ipsec_setup\(8\)**](#)). Here's an example:

```
config setup

interfaces="ipsec0=eth1 ipsec1=ppp0"

klipsdebug=none

plutodebug=control

protostack=auto

manualstart=
```

Parameters are optional unless marked "(required)".

The currently-accepted *parameter* names in a **config setup** section are:

myid

the identity to be used for **%myid**. **%myid** is used in the implicit policy group conns and can be used as an identity in explicit conns. If unspecified, **%myid** is set to the IP address in **%defaultroute** (if that is supported by a TXT record in its reverse domain), or otherwise the system's hostname (if that is supported by a TXT record in its forward domain), or otherwise it is undefined. An explicit value generally starts with "@".

protostack

decide which protocol stack is going to be used. Valid values are "auto", "klips", "netkey" and "mast". The "mast" stack is a variation for the klips stack.

interfaces

virtual and physical interfaces for IPsec to use: a single *virtual=physical* pair, a (quoted!) list of pairs separated by white space, or **%none**. One of the pairs may be written as **%defaultroute**, which means: find the interface *d* that the default route points to, and then act as if the value was "**ipsec0=d**". **%defaultroute** is the default; **%none** must be used to denote no interfaces, or when using the NETKEY stack. If **%defaultroute** is used (implicitly or explicitly) information about the default route and its interface is noted for use by [**ipsec_manual\(8\)**](#) and [**ipsec_auto\(8\)**](#).)

listen

IP address to listen on (default depends on **interfaces=** setting). Currently only accepts one IP address.

nat_traversal

whether to accept/offer to support NAT (NAPT, also known as "IP Masquerade") workaround for IPsec. Acceptable values are: **yes** and **no** (the default). This parameter may eventually become per-connection.

disable_port_floating

whether to enable the newer NAT-T standards for port floating. Acceptable values are **no** (the default) and **yes**.

force_keepalive

whether to force sending NAT-T keep-alives to support NAT which are sent to prevent the NAT router from closing its port when there is not enough traffic on the IPsec connection. Acceptable values are: **yes** and **no** (the default). This parameter may eventually become per-connection.

keep_alive

The delay (in seconds) for NAT-T keep-alive packets, if these are enabled using **force_keepalive**. This parameter may eventually become per-connection.

virtual_private

contains the networks that are allowed as subnet= for the remote client. In other words, the address ranges that may live behind a NAT router through which a client connects. This value is usually set to all the RFC-1918 address space, excluding the space used in the local subnet behind the NAT (An IP address cannot live at two places at once). IPv4 address ranges are denoted as %v4:a.b.c.d/mm and IPv6 is denoted as %v6:aaaa::bbbb:cccc:ddd:eeee/mm. One can exclude subnets by using the !. For example, if the VPN server is giving access to 192.168.1.0/24, this option should be set to:

`virtual_private=%v4:10.0.0.0/8,%v4:192.168.0.0/16,%v4:172.16.0.0/12,%v4:!192.168.1.0/24.`

This parameter is only needed on the server side and not on the client side that resides behind the NAT router, as the client will just use its IP address for the inner IP setting. This parameter may eventually become per-connection.

oe

a boolean (yes/no) that determines if Opportunistic Encryption will be enabled. Opportunistic Encryption is the term to describe using IPsec tunnels without prearrangement. It uses IPSECKEY or TXT records to announce public RSA keys for certain IP's or identities.

For a complete description see /doc/draft-richardson-ipsec-opportunistic.txt, doc/opportunism-spec.txt and doc/opportunism.howto. See also the IETF BTNS working group and RFC4025.

The default is no.

This feature is only available with kernel drivers that support the caching of packets (%hold eroutes or equivalent) that allows us to respond to a packet from an unknown IP address. At present only the (klips)masm protocol stack supports this feature.

nhelpers

how many *pluto helpers* are started to help with cryptographic operations. Pluto will start (*n*-1) of them, where *n* is the number of CPU's you have (including hyperthreaded CPU's). A value of 0 forces pluto to do all operations in the main process. A value of -1 tells pluto to perform the above calculation. Any other value forces the number to that amount.

crlcheckinterval

interval, specified in seconds, after which pluto will verify loaded X.509 CRL's for expiration. If any of the CRL's is expired, or if they previously failed to get updated, a new attempt at updating the CRL is made. The first attempt to update a CRL is started at two times the crlcheckinterval. If set to 0, which is also the default value if this option is not specified, CRL updating is disabled.

strictcrlpolicy

if not set, pluto is tolerant about missing or expired X.509 Certificate Revocation Lists (CRL's), and will allow peer certificates as long as they do not appear on an expired CRL. When this option is enabled, all connections with an expired or missing CRL will be denied. Active connections will be terminated at rekey time. This setup is more secure, but also dangerous. If the CRL is fetched through an IPsec tunnel with a CRL that expired, the entire VPN server will be dead in the water until a new CRL is manually transferred to the machine (if it allows non-IPsec connections).

Acceptable values are **yes** or **no** (the default).

forwardcontrol

This option is obsolete and ignored. Please use **net.ipv4.ip_forward = 0** in /etc/sysctl.conf instead to control the ip forwarding behaviour.

rp_filter

This option is obsolete and ignored. Please use the **net.ipv4.conf/[iface]/rp_filter = 0** options in /etc/sysctl.conf instead. This option is badly documented; it must be 0 in many cases for ipsec to function.

syslog

the **syslog**(2) "facility" name and priority to use for startup/shutdown log messages, default **daemon.error**.

klipsdebug

how much KLIPS debugging output should be logged. An empty value, or the magic value **none**, means no debugging output (the default). The magic value **all** means full output. Otherwise only the specified types of output (a quoted list, names separated by white space) are enabled; for details on available debugging types, see **ipsec_klipsdebug**(8). This KLIPS option has no effect on NETKEY, Windows or BSD stacks.

plutodebug

how much Pluto debugging output should be logged. An empty value, or the magic value **none**, means no debugging output (the default). The magic value **all** means full output. Otherwise only the specified types of output (a quoted list, names without the **--debug-** prefix, separated by white space) are enabled; for details on available debugging types, see **ipsec_pluto**(8).

uniqueids

whether a particular participant ID should be kept unique, with any new (automatically keyed) connection using an ID from a different IP address deemed to replace all old ones using that ID. Acceptable values are **yes** (the default) and **no**. Participant IDs normally are unique, so a new (automatically-keyed) connection using the same ID is almost invariably intended to replace an old one.

plutorestartoncrash

prevent pluto from restarting after it crashed. This option should only be used when debugging a crasher. It will prevent overwriting a core file on a new start, or a cascade of core files. This option is also required if used with plutostderlog= to avoid clearing the logs of the crasher.

Values can be yes (the default) or no.

plutoopts

additional options to pass to pluto upon startup. See **ipsec_pluto**(8).

plutostderlog

do not use syslog, but rather log to stderr, and direct stderr to the argument file.

pluto

whether to start Pluto or not; Values are **yes** (the default) or **no** (useful only in special circumstances).

plutowait

should Pluto wait for each negotiation attempt that is part of startup to finish before proceeding with the next? Values are **yes** or **no** (the default).

prepluto

shell command to run before starting Pluto (e.g., to decrypt an encrypted copy of the *ipsec.secrets* file). It's run in a very simple way; complexities like I/O redirection are best hidden within a script. Any output is redirected for logging, so running interactive commands is difficult unless they use /dev/tty or equivalent for their interaction. Default is none.

postpluto

shell command to run after starting Pluto (e.g., to remove a decrypted copy of the *ipsec.secrets* file). It's run in a very simple way; complexities like I/O redirection are best hidden within a script. Any output is redirected for logging, so running interactive commands is difficult unless they use /dev/tty or equivalent for their interaction. Default is none.

dumpdir

in what directory should things started by *setup* (notably the Pluto daemon) be allowed to dump core? The empty value (the default) means they are not allowed to.

fragicmp

whether a tunnel's need to fragment a packet should be reported back with an ICMP message, in an attempt to make the sender lower his PMTU estimate; acceptable values are **yes** (the default) and **no**. This KLIPS option has no effect on NETKEY, Windows or BSD stacks.

hidetos

whether a tunnel packet's TOS field should be set to 0 rather than copied from the user packet inside; acceptable values are **yes** (the default) and **no**. This KLIPS option has no effect on NETKEY, Windows or BSD stacks.

overridemtu

value that the MTU of the ipsecn **interface(s)** should be set to, overriding IPsec's (large) default. This parameter is needed only in special situations. This KLIPS option has no effect on NETKEY, Windows or BSD stacks.

Implicit Conns

The system automatically defines several conns to implement default policy groups. Each can be overridden by explicitly defining a new conn with the same name. If the new conn has **auto=ignore**, the definition is suppressed.

Here are the automatically supplied definitions.

```
conn clear
type=passthrough
authby=never
left=%defaultroute
right=%group
auto=route
conn clear-or-private
type=passthrough
left=%defaultroute
leftid=%myid
```

right=%opportunisticgroup

failureshunt=passthrough

keyingtries=3

ikelifetime=1h

salifetime=1h

rekey=no

auto=route

conn private-or-clear

type=tunnel

left=%defaultroute

leftid=%myid

right=%opportunisticgroup

failureshunt=passthrough

keyingtries=3

ikelifetime=1h

salifetime=1h

rekey=no

auto=route

conn private

type=tunnel

left=%defaultroute

leftid=%myid

right=%opportunisticgroup

failureshunt=drop

keyingtries=3

ikelifetime=1h

salifetime=1h

rekey=no

auto=route

conn block

```
type=reject
```

```
authby=never
```

```
left=%defaultroute
```

```
right=%group
```

```
auto=route
```

```
# default policy
```

```
conn packetdefault
```

```
type=tunnel
```

```
left=%defaultroute
```

```
leftid=%myid
```

```
left=0.0.0.0/0
```

```
right=%opportunistic
```

```
failureshunt=passthrough
```

```
keyingtries=3
```

```
ikelifetime=1h
```

```
salifetime=1h
```

```
rekey=no
```

```
auto=route
```

These conns are *not* affected by anything in **conn %default**. They will only work if **%defaultroute** works. The **leftid** will be the interfaces IP address; this requires that reverse DNS records be set up properly.

The implicit conns are defined after all others. It is appropriate and reasonable to use **also=private-or-clear** (for example) in any other opportunistic conn.

Policy Group Files

The optional files under /etc/ipsec.d/policy, including

```
/etc/ipsec.d/policies/clear
/etc/ipsec.d/policies/clear-or-private
/etc/ipsec.d/policies/private-or-clear
/etc/ipsec.d/policies/private
/etc/ipsec.d/policies/block
```

may contain policy group configuration information to supplement *ipsec.conf*. Their contents are not security-sensitive.

These files are text files. Each consists of a list of CIDR blocks, one per line. White space followed by # followed by anything to the end of the line is a comment and is ignored, as are empty lines.

A connection in ipsec.conf which has **right=%group** or **right=%opportunisticgroup** is a policy group connection. When a policy group file of the same name is loaded, with

ipsec auto --rereadgroups

or at system start, the connection is instantiated such that each CIDR block serves as an instance's **right** value. The system treats the resulting instances as normal connections.

For example, given a suitable connection definition **private**, and the file /etc/ipsec.d/policy/private with an entry 192.0.2.3, the system creates a connection instance **private#192.0.2.3**. This connection inherits all details from **private**, except that its right client is 192.0.2.3.

Default Policy Groups

The standard Openswan install includes several policy groups which provide a way of classifying possible peers into IPsec security classes: **private** (talk encrypted only), **private-or-clear** (prefer encryption), **clear-or-private** (respond to requests for encryption), **clear** and **block**. Implicit policy groups apply to the local host only, and are implemented by the **IMPLICIT CONNECTIONS** described above.

CHOOSING A CONNECTION [THIS SECTION IS EXTREMELY OUT OF DATE]

When choosing a connection to apply to an outbound packet caught with a **%trap**, the system prefers the one with the most specific eroute that includes the packet's source and destination IP addresses. Source subnets are examined before destination subnets. For initiating, only routed connections are considered. For responding, unrouted but added connections are considered.

When choosing a connection to use to respond to a negotiation which doesn't match an ordinary conn, an opportunistic connection may be instantiated. Eventually, its instance will be /32 -> /32, but for earlier stages of the negotiation, there will not be enough information about the client subnets to complete the instantiation.

Files

```
/etc/ipsec.conf
/etc/ipsec.d/policies/clear
/etc/ipsec.d/policies/clear-or-private
/etc/ipsec.d/policies/private-or-clear
/etc/ipsec.d/policies/private
/etc/ipsec.d/policies/block
```

See Also

[**ipsec**\(8\)](#), [**ipsec_ttoaddr**\(8\)](#), [**ipsec_auto**\(8\)](#), [**ipsec_manual**\(8\)](#), [**ipsec_rsasigkey**\(8\)](#)

History

Designed for the FreeS/WAN project <<http://www.freeswan.org>> by Henry Spencer.

Bugs

Before reporting new bugs, please ensure you are using the latest version of Openswan, and if not using KLIPS, please ensure you are using the latest kernel code for your IPsec stack.

When **type** or **failureshunt** is set to **drop** or **reject**, Openswan blocks outbound packets using eroutes, but assumes inbound blocking is handled by the firewall. Openswan offers firewall hooks via an "updown" script. However, the default **ipsec_updown** provides no help in controlling a modern firewall.

Including attributes of the keying channel (authentication methods, **ikelifetime**, etc.) as an attribute of a connection, rather than of a participant pair, is dubious and incurs limitations.

The use of **%any** with the **protoport=** option is ambiguous. Should the SA permits any port through or should the SA negotiate any single port through? The first is a basic conn with a wildcard. The second is a template. The second is the current behaviour, and it's wrong for quite a number of uses involving TCP. The keyword **%one** may be introduced in the future to separate these two cases.

ipsec_manual is not nearly as generous about the syntax of subnets, addresses, etc. as the usual Openswan user interfaces. Four-component dotted-decimal must be used for all addresses. It *is* smart enough to translate bit-count netmasks to dotted-decimal form.

It would be good to have a line-continuation syntax, especially for the very long lines involved in RSA signature keys.

First packet caching is only implemented for the KLIPS(NG) and MAST stacks. NETKEY returns POSIX-breaking responses, visible as *connect: Resource temporarily unavailable* errors. This affects Opportunistic Encryption and DPD. Functionality on the BSD and Windows stacks is unknown.

Some state information is only available when using KLIPS, and will return errors on other IPsec stacks. These include *ipsec eroute*, *ipsec spi* and *ipsec look*.

Multiple L2TP clients behind the same NAT router, and multiple L2TP clients behind different NAT routers using the same Virtual IP is currently only working for the KLIPSNG stack.

The ability to specify different identities, **authby**, and public keys for different automatic-keyed connections between the same participants is misleading; this doesn't work dependably because the identity of the participants is not known early enough. This is especially awkward for the "Road Warrior" case, where the remote IP address is specified as 0.0.0.0, and that is considered to be the "participant" for such connections.

In principle it might be necessary to control MTU on an interface-by-interface basis, rather than with the single global override that **overridemu** provides. This feature is planned for a future release.

A number of features which *could* be implemented in both manual and automatic keying actually are not yet implemented for manual keying. This is unlikely to be fixed any time soon.

If conns are to be added before DNS is available, **left=FQDN**, **leftnexttop=FQDN**, and **leftrsasigkey=%dns.onload** will fail. **ipsec_pluto**(8) does not actually use the public key for our side of a conn but it isn't generally known at a add-time which side is ours (Road Warrior and Opportunistic conns are currently exceptions).

The **myid** option does not affect explicit **ipsec auto --add** or **ipsec auto --replace** commands for implicit conn.

Referenced By

[**ipsec_ipsec_pluto**\(8\)](#), [**ipsec_showhostkey**\(8\)](#), [**strongswan**\(8\)](#), [**strongswan.conf**\(5\)](#),
[**strongswan_ipsec.secrets**\(5\)](#), [**strongswan_openac**\(8\)](#), [**strongswan_pluto**\(8\)](#)

ipsec__updown(8) - Linux man page

Name

ipsec__updown - kernel and routing manipulation script

Synopsis

_updown is invoked by pluto when it has brought up a new connection. This script is used to insert the appropriate routing entries for IPsec operation on some kernel IPsec stacks, such as KLIPS and MAST, and may do other necessary work that is kernel or user specific, such as defining custom firewall rules. The interface to the script is documented in the pluto man page.

Variables

The *_updown* is passed along a number of variables which can be used to act differently based on the information:

PLUTO_VERSION

indicates what version of this interface is being used. This document describes version 1.1. This is upwardly compatible with version 1.0.

PLUTO_VERB

specifies the name of the operation to be performed, which can be one of **prepare-host**, **prepare-client**, **up-host**, **up-client**, **down-host** or **down-client**. If the address family for security gateway to security gateway communications is IPv6, then a suffix of -v6 is added to this verb.

PLUTO_CONNECTION

is the name of the connection for which we are routing.

PLUTO_NEXT_HOP

is the next hop to which packets bound for the peer must be sent.

PLUTO_INTERFACE

is the name of the ipsec interface to be used.

PLUTO_ME

is the IP address of our host.

PLUTO_MY_CLIENT

is the IP address / count of our client subnet. If the client is just the host, this will be the host's own IP address / max (where max is 32 for IPv4 and 128 for IPv6).

PLUTO_MY_CLIENT_NET

is the IP address of our client net. If the client is just the host, this will be the host's own IP address.

PLUTO_MY_CLIENT_MASK

is the mask for our client net. If the client is just the host, this will be 255.255.255.255.

PLUTO_PEER

is the IP address of our peer.

PLUTO_PEER_CLIENT

is the IP address / count of the peer's client subnet. If the client is just the peer, this will be the peer's own IP address / max (where max is 32 for IPv4 and 128 for IPv6).

PLUTO_PEER_CLIENT_NET

is the IP address of the peer's client net. If the client is just the peer, this will be the peer's own IP address.

PLUTO_PEER_CLIENT_MASK

is the mask for the peer's client net. If the client is just the peer, this will be 255.255.255.255.

PLUTO_MY_PROTOCOL

lists the protocols allowed over this IPsec SA.

PLUTO_PEER_PROTOCOL

lists the protocols the peer allows over this IPsec SA.

PLUTO_MY_PORT

lists the ports allowed over this IPsec SA.

PLUTO_PEER_PORT

lists the ports the peer allows over this IPsec SA.

PLUTO_MY_ID

lists our id.

PLUTO_PEER_ID

lists our peer's id.

PLUTO_PEER_CA

lists the peer's CA.

See Also

[ipsec\(8\)](#), [ipsec_pluto\(8\)](#).

History

Man page written for the Linux FreeS/WAN project <<http://www.freeswan.org/>> by Michael Richardson. Original program written by Henry Spencer.

ipsec_policy(8) - Linux man page

Name

ipsec_policy - show ipsec policy information

Synopsis

```
# detect what stack is used
ipsec policy --detect-stack

# display policy information
ipsec policy [ --all | [ --inbound | --outbound | --forward ] ] \
    [ --stack=name ] [ --read=file ] [ --debug ]

# provide usage information
ipsec policy --usage
ipsec policy --help
```

Description

policy displays the incoming, outgoing, and forwarding packet policies of the system. It is a wrapper around existing klips and netkey data, but presented in a less terse form.

Options

--detect-stack

Only display the stack that Openswan is using. Possible results are.

klips

KLIPS is the Openswan ipsec kernel module. This stack type indicates that KLIPS is not running in *mast* mode (see next option), but rather in the default mode. In this mode, KLIPS outgoing packet policy is dictated by *eroutes*. See the **ipsec_eroute** man page for further details.

mast

This is a mode of the Openswan ipsec kernel module, KLIPS . In this mode outgoing packet routing policies are dictated by iptables, and Linux kernel policy routing. This mode is selected by using "protostack=mast" setting in *ipsec.conf*.

netkey

This stack indicates that Openswan is controlling the Linux kernel built-in ipsec functionally.

--all

Show inbound, outbound, and forward policies. This is the default.

--inbound --in

Show only inbound policy.

--outbound --out

 Show only outbound policy.

--forward --fwd

 Show only forward policy.

--stack=<name>

 Skip autodetection and force read policy from this stack. See help on **--detect-stack** (above) for valid options and their descriptions.

--read=<file>

 This option overrides what file would be read to gather the policy information. It could be used to read policy information from a snapshot obtained from a running system.

In the case of the klips or mast stack, this file is the output of the

/proc/net/ipsec/spi/all file.

--help

 Output help.

--debug

 Output debug info.

Files

/proc/net/ipsec/spi/all

See Also

[**ipsec**\(8\)](#), [**ipsec_eroute**\(8\)](#), [**ipsec_manual**\(8\)](#)

History

Designed for the Openswan project <<http://www.openswan.org>> by Bart Trojanowski.

Bugs

Does not support netkey yet.

ipsec_selinux(8) - Linux man page

Name

ipsec_selinux - Security Enhanced Linux Policy for the ipsec processes

Description

Security-Enhanced Linux secures the ipsec processes via flexible mandatory access control.

The ipsec processes execute with the `ipsec_t` SELinux type. You can check if you have these processes running by executing the `ps` command with the `-Z` qualifier.

For example:

```
ps -eZ | grep ipsec_t
```

Entrypoints

The `ipsec_t` SELinux type can be entered via the "ipsec_exec_t" file type. The default entrypoint paths for the `ipsec_t` domain are the following:"

```
/usr/lib(64)?/ipsec/spi, /usr/lib(64)?/ipsec/pluto, /usr/lib(64)?/ipsec/eroute,  
/usr/lib(64)?/ipsec/klipsdebug, /usr/local/lib(64)?/ipsec/spi,  
/usr/local/lib(64)?/ipsec/pluto, /usr/local/lib(64)?/ipsec/eroute,  
/usr/local/lib(64)?/ipsec/klipsdebug, /usr/libexec/ipsec/spi, /usr/libexec/ipsec/pluto,  
/usr/libexec/ipsec/eroute, /usr/libexec/ipsec/klipsdebug
```

Process Types

SELinux defines process types (domains) for each process running on the system

You can see the context of a process using the `-Z` option to `ps`

Policy governs the access confined processes have to files. SELinux ipsec policy is very flexible allowing users to setup their ipsec processes in as secure a method as possible.

The following process types are defined for ipsec:

ipsec_t, ipsec_mgmt_t

Note: **semanage permissive -a ipsec_t**

can be used to make the process type `ipsec_t` permissive. Permissive process types are not denied access by SELinux. AVC messages will still be generated.

File Contexts

SELinux requires files to have an extended attribute to define the file type.

You can see the context of a file using the **-Z** option to **ls**

Policy governs the access confined processes have to these files. SELinux ipsec policy is very flexible allowing users to setup their ipsec processes in as secure a method as possible.

The following file types are defined for ipsec:

ipsec_conf_file_t

- Set files with the ipsec_conf_file_t type, if you want to treat the files as ipsec conf content.

ipsec_exec_t

- Set files with the ipsec_exec_t type, if you want to transition an executable to the ipsec_t domain.

ipsec_initrc_exec_t

- Set files with the ipsec_initrc_exec_t type, if you want to transition an executable to the ipsec_initrc_t domain.

ipsec_key_file_t

- Set files with the ipsec_key_file_t type, if you want to treat the files as ipsec key content.

ipsec_log_t

- Set files with the ipsec_log_t type, if you want to treat the data as ipsec log data, usually stored under the /var/log directory.

ipsec_mgmt_exec_t

- Set files with the ipsec_mgmt_exec_t type, if you want to transition an executable to the ipsec_mgmt_t domain.

ipsec_mgmt_lock_t

- Set files with the ipsec_mgmt_lock_t type, if you want to treat the files as ipsec mgmt lock data, stored under the /var/lock directory

ipsec_mgmt_var_run_t

- Set files with the ipsec_mgmt_var_run_t type, if you want to store the ipsec mgmt files under the /run directory.

ipsec_tmp_t

- Set files with the ipsec_tmp_t type, if you want to store ipsec temporary files in the /tmp directories.

ipsec_var_run_t

- Set files with the ipsec_var_run_t type, if you want to store the ipsec files under the /run directory.

Note: File context can be temporarily modified with the chcon command. If you want to permanently change the file context you need to use the **semanage fcontext** command. This will modify the SELinux labeling database. You will need to use **restorecon** to apply the labels.

Port Types

SELinux defines port types to represent TCP and UDP ports.

You can see the types associated with a port by using the following command:

semanage port -l

Policy governs the access confined processes have to these ports. SELinux ipsec policy is very flexible allowing users to setup their ipsec processes in as secure a method as possible.

The following port types are defined for ipsec:

ipsecnat_port_t

Default Defined Ports:

tcp 4500 udp 4500

Managed Files

The SELinux process type ipsec_t can manage files labeled with the following file types. The paths listed are the default paths for these file types. Note the processes UID still need to have DAC permissions.

initrc_tmp_t

ipsec_key_file_t

/etc/ipsec.d(/.*)?

/etc/racoon/certs(/.*)?

/etc/ipsec.secrets

/etc/racoon/psk.txt

ipsec_tmp_t

ipsec_var_run_t

/var/racoon(/.*)?

/var/run/pluto(/.*)?

/var/run/racoon.pid

mnt_t

/mnt(/[^/]*)

/mnt(/[^/]*)?

/rhev(/[^/]*)?

/media(/[^/]*)

/media(/[^/]*)?

/etc/rhgb(/.*)?

/media/.hal-.*

/net

/afs

/misc

/rhev

net_conf_t

/etc/ntp??.conf.*

/etc/yp.conf.*

/etc/denyhosts.*

/etc/hosts.deny.*

/etc/resolv.conf.*

/etc/ntp/step-tickers.*

```
/etc/sysconfig/networking(.*)?  
/etc/sysconfig/network-scripts(.*)?  
/etc/sysconfig/network-scripts/*resolv.conf  
/etc/hosts  
/etc/ethers  
root_t  
/  
/initrd  
security_t  
tmp_t  
/tmp  
/usr/tmp  
/var/tmp  
/var/tmp/vi.recover
```

Commands

semanage fcontext can also be used to manipulate default file context mappings.

semanage permissive can also be used to manipulate whether or not a process type is permissive.

semanage module can also be used to enable/disable/install/remove policy modules.

semanage port can also be used to manipulate the port definitions

system-config-selinux is a GUI tool available to customize SELinux policy settings.

Author

This manual page was auto-generated using **sepolicy manpage** by mgrepl.

See Also

selinux(8), **ipsec(8)**, **semanage(8)**, **restorecon(8)**, **chcon(1)**, **sepolicy(8)** ,
ipsec_mgmt_selinux(8)

NAME

iptables-apply – a safer way to update iptables remotely

SYNOPSIS

iptables-apply [-hV] [-t timeout] [-w savefile] {[rulesfile]/-c [runcmd]}

DESCRIPTION

iptables-apply will try to apply a new rulesfile (as output by iptables-save, read by iptables-restore) or run a command to configure iptables and then prompt the user whether the changes are okay. If the new iptables rules cut the existing connection, the user will not be able to answer affirmatively. In this case, the script rolls back to the previous working iptables rules after the timeout expires.

Successfully applied rules can also be written to savefile and later used to roll back to this state. This can be used to implement a store last good configuration mechanism when experimenting with an iptables setup script: iptables-apply -w /etc/network/iptables.up.rules -c /etc/network/iptables.up.run

When called as ip6tables-apply, the script will use ip6tables-save/-restore and IPv6 default values instead. Default value for rulesfile is '/etc/network/iptables.up.rules'.

OPTIONS

-t seconds, --timeout seconds

Sets the timeout in seconds after which the script will roll back to the previous ruleset (default: 10).

-w savefile, --write savefile

Specify the savefile where successfully applied rules will be written to (default if empty string is given: /etc/network/iptables.up.rules).

-c runcmd, --command runcmd

Run command runcmd to configure iptables instead of applying a rulesfile (default: /etc/network/iptables.up.run).

-h, --help

Display usage information.

-V, --version

Display version information.

SEE ALSO

iptables-restore(8), iptables-save(8), iptables(8).

LEGALESE

Original iptables-apply - Copyright 2006 Martin F. Krafft <madduck@madduck.net>. Version 1.1 - Copyright 2010 GW <gw.2010@tnode.com or http://gw.tnode.com/>.

This manual page was written by Martin F. Krafft <madduck@madduck.net> and extended by GW <gw.2010@tnode.com or http://gw.tnode.com/>.

Permission is granted to copy, distribute and/or modify this document under the terms of the Artistic License 2.0.

NAME

xtables-nft — iptables using nftables kernel api

DESCRIPTION

xtables-nft are versions of iptables that use the nftables API. This is a set of tools to help the system administrator migrate the ruleset from **iptables(8)**, **ip6tables(8)**, **arptables(8)**, and **ebtables(8)** to **nftables(8)**.

The **xtables-nft** set is composed of several commands:

- **iptables-nft**
- **iptables-nft-save**
- **iptables-nft-restore**
- **ip6tables-nft**
- **ip6tables-nft-save**
- **ip6tables-nft-restore**
- **arptables-nft**
- **ebtables-nft**

These tools use the libxtables framework extensions and hook to the nf_tables kernel subsystem using the **nft_compat** module.

USAGE

The xtables-nft tools allow you to manage the nf_tables backend using the native syntax of **iptables(8)**, **ip6tables(8)**, **arptables(8)**, and **ebtables(8)**.

You should use the xtables-nft tools exactly the same way as you would use the corresponding original tools.

Adding a rule will result in that rule being added to the nf_tables kernel subsystem instead. Listing the ruleset will use the nf_tables backend as well.

When these tools were designed, the main idea was to replace each legacy binary with a symlink to the xtables-nft program, for example:

```
/sbin/iptables -> /usr/sbin/iptables-nft-multi
/sbin/ip6tables -> /usr/sbin/ip6tables-nft-multi
/sbin/arptables -> /usr/sbin/arptables-nft-multi
/sbin/ebtables -> /usr/sbin/ebtables-nft-multi
```

The iptables version string will indicate whether the legacy API (get/setsockopt) or the new nf_tables api is used:

```
iptables -V
iptables v1.7 (nf_tables)
```

DIFFERENCES TO LEGACY IPTABLES

Because the xtables-nft tools use the nf_tables kernel API, rule additions and deletions are always atomic. Unlike iptables-legacy, iptables-nft -A ... will NOT need to retrieve the current ruleset from the kernel, change it, and re-load the altered ruleset. Instead, iptables-nft will tell the kernel to add one rule. For this reason, the iptables-legacy --wait option is a no-op in iptables-nft.

Use of the xtables-nft tools allow monitoring ruleset changes using the **xtables-monitor(8)** command.

When using `-j TRACE` to debug packet traversal to the ruleset, note that you will need to use **xtables-monitor(8)** in `--trace` mode to obtain monitoring trace events.

EXAMPLES

One basic example is creating the skeleton ruleset in nf_tables from the xtables-nft tools, in a fresh machine:

```
root@machine:~# iptables-nft -L
[...]
root@machine:~# ip6tables-nft -L
[...]
root@machine:~# arptables-nft -L
[...]
root@machine:~# ebtables-nft -L
[...]
root@machine:~# nft list ruleset
table ip filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}
table ip6 filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}
table bridge filter {
    chain INPUT {
        type filter hook input priority -200; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority -200; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority -200; policy accept;
    }
}
```

```

}
table arp filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy accept;
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept;
    }
}

```

(please note that in fresh machines, listing the ruleset for the first time results in all tables an chain being created).

To migrate your complete filter ruleset, in the case of **iptables(8)**, you would use:

```

root@machine:~# iptables-legacy-save > myruleset # reads from x_tables
root@machine:~# iptables-nft-restore myruleset # writes to nf_tables
or
root@machine:~# iptables-legacy-save | iptables-translate-restore | less

```

to see how rules would look like in the nft **nft(8)** syntax.

LIMITATIONS

You should use **Linux kernel >= 4.17**.

The CLUSTERIP target is not supported.

To get up-to-date information about this, please head to <http://wiki.nftables.org/>.

SEE ALSO

nft(8), xtables-translate(8), xtables-monitor(8)

AUTHORS

The nftables framework is written by the Netfilter project (<https://www.netfilter.org>).

This manual page was written by Arturo Borrero Gonzalez <arturo@debian.org> for the Debian project, but may be used by others.

This documentation is free/libre under the terms of the GPLv2+.

NAME

iptables/**ip6tables** — administration tool for IPv4/IPv6 packet filtering and NAT

SYNOPSIS

```
iptables [-t table] {-A|-C|-D} chain rule-specification
ip6tables [-t table] {-A|-C|-D} chain rule-specification
iptables [-t table] -I chain [rulenumber] rule-specification
iptables [-t table] -R chain rulenumber rule-specification
iptables [-t table] -D chain rulenumber
iptables [-t table] -S [chain [rulenumber]]
iptables [-t table] {-F|-L|-Z} [chain [rulenumber]] [options...]
iptables [-t table] -N chain
iptables [-t table] -X [chain]
iptables [-t table] -P chain target
iptables [-t table] -E old-chain-name new-chain-name
rule-specification = [matches...] [target]
match = -m matchname [per-match-options]
target = -j targetname [per-target-options]
```

DESCRIPTION

Iptables and **ip6tables** are used to set up, maintain, and inspect the tables of IPv4 and IPv6 packet filter rules in the Linux kernel. Several different tables may be defined. Each table contains a number of built-in chains and may also contain user-defined chains.

Each chain is a list of rules which can match a set of packets. Each rule specifies what to do with a packet that matches. This is called a ‘target’, which may be a jump to a user-defined chain in the same table.

TARGETS

A firewall rule specifies criteria for a packet and a target. If the packet does not match, the next rule in the chain is examined; if it does match, then the next rule is specified by the value of the target, which can be the name of a user-defined chain, one of the targets described in **iptables-extensions(8)**, or one of the special values **ACCEPT**, **DROP** or **RETURN**.

ACCEPT means to let the packet through. **DROP** means to drop the packet on the floor. **RETURN** means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target **RETURN** is matched, the target specified by the chain policy determines the fate of the packet.

TABLES

There are currently five independent tables (which tables are present at any time depends on the kernel configuration options and which modules are present).

-t, --table table

This option specifies the packet matching table which the command should operate on. If the kernel is configured with automatic module loading, an attempt will be made to load the appropriate module for that table if it is not already there.

The tables are as follows:

filter: This is the default table (if no **-t** option is passed). It contains the built-in chains **INPUT** (for packets destined to local sockets), **FORWARD** (for packets being routed through the box), and **OUTPUT** (for locally-generated packets).

nat: This table is consulted when a packet that creates a new connection is encountered. It consists of four built-ins: **PREROUTING** (for altering packets as soon as they come in),

INPUT (for altering packets destined for local sockets), **OUTPUT** (for altering locally-generated packets before routing), and **POSTROUTING** (for altering packets as they are about to go out). IPv6 NAT support is available since kernel 3.7.

mangle:

This table is used for specialized packet alteration. Until kernel 2.4.17 it had two built-in chains: **PREROUTING** (for altering incoming packets before routing) and **OUTPUT** (for altering locally-generated packets before routing). Since kernel 2.4.18, three other built-in chains are also supported: **INPUT** (for packets coming into the box itself), **FORWARD** (for altering packets being routed through the box), and **POSTROUTING** (for altering packets as they are about to go out).

raw: This table is used mainly for configuring exemptions from connection tracking in combination with the NOTRACK target. It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables. It provides the following built-in chains: **PREROUTING** (for packets arriving via any network interface) **OUTPUT** (for packets generated by local processes)

security:

This table is used for Mandatory Access Control (MAC) networking rules, such as those enabled by the **SECMARK** and **CONNSECMARK** targets. Mandatory Access Control is implemented by Linux Security Modules such as SELinux. The security table is called after the filter table, allowing any Discretionary Access Control (DAC) rules in the filter table to take effect before MAC rules. This table provides the following built-in chains: **INPUT** (for packets coming into the box itself), **OUTPUT** (for altering locally-generated packets before routing), and **FORWARD** (for altering packets being routed through the box).

OPTIONS

The options that are recognized by **iptables** and **ip6tables** can be divided into several different groups.

COMMANDS

These options specify the desired action to perform. Only one of them can be specified on the command line unless otherwise stated below. For long versions of the command and option names, you need to use only enough letters to ensure that **iptables** can differentiate it from all other options.

-A, --append *chain rule-specification*

Append one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.

-C, --check *chain rule-specification*

Check whether a rule matching the specification does exist in the selected chain. This command uses the same logic as **-D** to find a matching entry, but does not alter the existing iptables configuration and uses its exit code to indicate success or failure.

-D, --delete *chain rule-specification*

-D, --delete *chain rulenum*

Delete one or more rules from the selected chain. There are two versions of this command: the rule can be specified as a number in the chain (starting at 1 for the first rule) or a rule to match.

-I, --insert *chain [rulenum] rule-specification*

Insert one or more rules in the selected chain as the given rule number. So, if the rule number is 1, the rule or rules are inserted at the head of the chain. This is also the default if no rule number is specified.

-R, --replace *chain rulenum rule-specification*

Replace a rule in the selected chain. If the source and/or destination names resolve to multiple addresses, the command will fail. Rules are numbered starting at 1.

-L, --list [chain]

List all rules in the selected chain. If no chain is selected, all chains are listed. Like every other iptables command, it applies to the specified table (filter is the default), so NAT rules get listed by `iptables -t nat -n -L`

Please note that it is often used with the `-n` option, in order to avoid long reverse DNS lookups. It is legal to specify the `-Z` (zero) option as well, in which case the chain(s) will be atomically listed and zeroed. The exact output is affected by the other arguments given. The exact rules are suppressed until you use

`iptables -L -v`
or `iptables-save(8)`.

-S, --list-rules [chain]

Print all rules in the selected chain. If no chain is selected, all chains are printed like `iptables-save`. Like every other iptables command, it applies to the specified table (filter is the default).

-F, --flush [chain]

Flush the selected chain (all the chains in the table if none is given). This is equivalent to deleting all the rules one by one.

-Z, --zero [chain [rulenum]]

Zero the packet and byte counters in all chains, or only the given chain, or only the given rule in a chain. It is legal to specify the `-L, --list` (list) option as well, to see the counters immediately before they are cleared. (See above.)

-N, --new-chain chain

Create a new user-defined chain by the given name. There must be no target of that name already.

-X, --delete-chain [chain]

Delete the optional user-defined chain specified. There must be no references to the chain. If there are, you must delete or replace the referring rules before the chain can be deleted. The chain must be empty, i.e. not contain any rules. If no argument is given, it will attempt to delete every non-built-in chain in the table.

-P, --policy chain target

Set the policy for the built-in (non-user-defined) chain to the given target. The policy target must be either **ACCEPT** or **DROP**.

-E, --rename-chain old-chain new-chain

Rename the user specified chain to the user supplied name. This is cosmetic, and has no effect on the structure of the table.

-h Help. Give a (currently very brief) description of the command syntax.**PARAMETERS**

The following parameters make up a rule specification (as used in the add, delete, insert, replace and append commands).

-4, --ipv4

This option has no effect in `iptables` and `iptables-restore`. If a rule using the `-4` option is inserted with (and only with) `ip6tables-restore`, it will be silently ignored. Any other uses will throw an error. This option allows IPv4 and IPv6 rules in a single rule file for use with both `iptables-restore` and `ip6tables-restore`.

-6, --ipv6

If a rule using the `-6` option is inserted with (and only with) `iptables-restore`, it will be silently ignored. Any other uses will throw an error. This option allows IPv4 and IPv6 rules in a single rule file for use with both `iptables-restore` and `ip6tables-restore`. This option has no effect in `ip6tables` and `ip6tables-restore`.

[!] -p, --protocol protocol

The protocol of the rule or of the packet to check. The specified protocol can be one of **tcp**, **udp**, **udplite**, **icmp**, **icmpv6**, **esp**, **ah**, **sctp**, **mh** or the special keyword "**all**", or it can be a numeric

value, representing one of these protocols or a different one. A protocol name from /etc/protocols is also allowed. A "!" argument before the protocol inverts the test. The number zero is equivalent to **all**. "all" will match with all protocols and is taken as default when this option is omitted. Note that, in ip6tables, IPv6 extension headers except **esp** are not allowed. **esp** and **ipv6-nonext** can be used with Kernel version 2.6.11 or later. The number zero is equivalent to **all**, which means that you cannot test the protocol field for the value 0 directly. To match on a HBH header, even if it were the last, you cannot use **-p 0**, but always need **-m hbh**.

[!] **-s, --source** *address[/mask][,...]*

Source specification. *Address* can be either a network name, a hostname, a network IP address (with */mask*), or a plain IP address. Hostnames will be resolved once only, before the rule is submitted to the kernel. Please note that specifying any name to be resolved with a remote query such as DNS is a really bad idea. The *mask* can be either an ipv4 network mask (for iptables) or a plain number, specifying the number of 1's at the left side of the network mask. Thus, an iptables mask of 24 is equivalent to 255.255.255.0. A "!" argument before the address specification inverts the sense of the address. The flag **--src** is an alias for this option. Multiple addresses can be specified, but this will **expand to multiple rules** (when adding with **-A**), or will cause multiple rules to be deleted (with **-D**).

[!] **-d, --destination** *address[/mask][,...]*

Destination specification. See the description of the **-s** (source) flag for a detailed description of the syntax. The flag **--dst** is an alias for this option.

-m, --match *match*

Specifies a match to use, that is, an extension module that tests for a specific property. The set of matches make up the condition under which a target is invoked. Matches are evaluated first to last as specified on the command line and work in short-circuit fashion, i.e. if one extension yields false, evaluation will stop.

-j, --jump *target*

This specifies the target of the rule; i.e., what to do if the packet matches it. The target can be a user-defined chain (other than the one this rule is in), one of the special builtin targets which decide the fate of the packet immediately, or an extension (see **EXTENSIONS** below). If this option is omitted in a rule (and **-g** is not used), then matching the rule will have no effect on the packet's fate, but the counters on the rule will be incremented.

-g, --goto *chain*

This specifies that the processing should continue in a user specified chain. Unlike the **--jump** option return will not continue processing in this chain but instead in the chain that called us via **--jump**.

[!] **-i, --in-interface** *name*

Name of an interface via which a packet was received (only for packets entering the **INPUT**, **FORWARD** and **PREROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[!] **-o, --out-interface** *name*

Name of an interface via which a packet is going to be sent (for packets entering the **FORWARD**, **OUTPUT** and **POSTROUTING** chains). When the "!" argument is used before the interface name, the sense is inverted. If the interface name ends in a "+", then any interface which begins with this name will match. If this option is omitted, any interface name will match.

[!] **-f, --fragment**

This means that the rule only refers to second and further IPv4 fragments of fragmented packets. Since there is no way to tell the source or destination ports of such a packet (or ICMP type), such a packet will not match any rules which specify them. When the "!" argument precedes the **-f** flag, the rule will only match head fragments, or unfragmented packets. This option is IPv4 specific, it is not available in ip6tables.

-c, --set-counters *packets bytes*

This enables the administrator to initialize the packet and byte counters of a rule (during **INSERT**, **APPEND**, **REPLACE** operations).

OTHER OPTIONS

The following additional options can be specified:

-v, --verbose

Verbose output. This option makes the list command show the interface name, the rule options (if any), and the TOS masks. The packet and byte counters are also listed, with the suffix 'K', 'M' or 'G' for 1000, 1,000,000 and 1,000,000,000 multipliers respectively (but see the **-x** flag to change this). For appending, insertion, deletion and replacement, this causes detailed information on the rule or rules to be printed. **-v** may be specified multiple times to possibly emit more detailed debug statements.

-w, --wait [*seconds*]

Wait for the xtables lock. To prevent multiple instances of the program from running concurrently, an attempt will be made to obtain an exclusive lock at launch. By default, the program will exit if the lock cannot be obtained. This option will make the program wait (indefinitely or for optional *seconds*) until the exclusive lock can be obtained.

-W, --wait-interval *microseconds*

Interval to wait per each iteration. When running latency sensitive applications, waiting for the xtables lock for extended durations may not be acceptable. This option will make each iteration take the amount of time specified. The default interval is 1 second. This option only works with **-w**.

-n, --numeric

Numeric output. IP addresses and port numbers will be printed in numeric format. By default, the program will try to display them as host names, network names, or services (whenever applicable).

-x, --exact

Expand numbers. Display the exact value of the packet and byte counters, instead of only the rounded number in K's (multiples of 1000) M's (multiples of 1000K) or G's (multiples of 1000M). This option is only relevant for the **-L** command.

--line-numbers

When listing rules, add line numbers to the beginning of each rule, corresponding to that rule's position in the chain.

--modprobe=*command*

When adding or inserting rules into a chain, use *command* to load any necessary modules (targets, match extensions, etc).

LOCK FILE

iptables uses the */run/xtables.lock* file to take an exclusive lock at launch.

The **XTABLES_LOCKFILE** environment variable can be used to override the default setting.

MATCH AND TARGET EXTENSIONS

iptables can use extended packet matching and target modules. A list of these is available in the **iptables-extensions(8)** manpage.

DIAGNOSTICS

Various error messages are printed to standard error. The exit code is 0 for correct functioning. Errors which appear to be caused by invalid or abused command line parameters cause an exit code of 2, and other errors cause an exit code of 1.

BUGS

Bugs? What's this? ;-) Well, you might want to have a look at <http://bugzilla.netfilter.org/>

COMPATIBILITY WITH IPCHAINS

This **iptables** is very similar to ipchains by Rusty Russell. The main difference is that the chains **INPUT** and **OUTPUT** are only traversed for packets coming into the local host and originating from the local host respectively. Hence every packet only passes through one of the three chains (except loopback traffic, which involves both INPUT and OUTPUT chains); previously a forwarded packet would pass through all three.

The other main difference is that **-i** refers to the input interface; **-o** refers to the output interface, and both are available for packets entering the **FORWARD** chain.

The various forms of NAT have been separated out; **iptables** is a pure packet filter when using the default 'filter' table, with optional extension modules. This should simplify much of the previous confusion over the combination of IP masquerading and packet filtering seen previously. So the following options are handled differently:

-j MASQ
-M -S
-M -L

There are several other changes in iptables.

SEE ALSO

iptables-apply(8), **iptables-save(8)**, **iptables-restore(8)**, **iptables-extensions(8)**,

The packet-filtering-HOWTO details iptables usage for packet filtering, the NAT-HOWTO details NAT, the netfilter-extensions-HOWTO details the extensions that are not in the standard distribution, and the netfilter-hacking-HOWTO details the netfilter internals.

See <http://www.netfilter.org/>.

AUTHORS

Rusty Russell originally wrote iptables, in early consultation with Michael Neuling.

Marc Boucher made Rusty abandon ipnatctl by lobbying for a generic packet selection framework in iptables, then wrote the mangle table, the owner match, the mark stuff, and ran around doing cool stuff everywhere.

James Morris wrote the TOS target, and tos match.

Jozsef Kadlecik wrote the REJECT target.

Harald Welte wrote the ULOG and NFQUEUE target, the new libiptc, as well as the TTL, DSCP, ECN matches and targets.

The Netfilter Core Team is: Jozsef Kadlecik, Pablo Neira Ayuso, Eric Leblond, Florian Westphal and Arturo Borrero Gonzalez. Emeritus Core Team members are: Marc Boucher, Martin Josefsson, Yasuyuki Kozakai, James Morris, Harald Welte and Rusty Russell.

Man page originally written by Herve Eychenne <rv@wallfire.org>.

VERSION

This manual page applies to iptables/ip6tables 1.8.7.

NAME

iptunnel – Create and manage IP tunnels

SYNOPSIS

```
iptunnel { add | change | del | show } [ NAME ]
[ mode { ipip | gre | sit } ] [ remote ADDR ] [ local ADDR ]
[ [i|o]seq ] [ [i|o]key KEY ] [ [i|o]csum ]
[ ttl TTL ] [ tos TOS ] [ nopmtudisc ] [ dev PHYS_DEV ]
```

iptunnel -V | --version

Where: NAME := STRING

```
ADDR := { IP_ADDRESS | any }
TOS := { NUMBER | inherit }
TTL := { 1..255 | inherit }
KEY := { DOTTED_QUAD | NUMBER }
```

DESCRIPTION

iptunnel can be used to add, change, delete and show IP tunnels on the machine.

OPTIONS

help Show the help text.

show List existing IP tunnels.

{ **add** | **change** | **del** } [NAME]

Specify an action (**add**, **change** or **delete** an IP tunnel) to be executed by **iptunnel** on the tunnel named **NAME**, which must be a string. You must specify one action.

mode PROTOCOL

Specify the **PROTOCOL** to be used for the IP tunnel. It can be **ipip** for TCP/IP, **gre** for Cisco GRE tunnel (RFCs 1701 and 1702), or **sit** for IPv6-in-IPv4 tunneling.

remote ADDR

Set the remote (i.e., destination) address of the tunnel. **ADDR** must be an IP address or the word **any**.

local ADDR

Set the local (i.e., source) address of the tunnel. **ADDR** must be an IP address or the word **any**.

iseq Require that all incoming packets are serialized. Only applicable for **GRE** tunnels.

oseq Enable serialization (sequencing) for all outgoing packets. Only applicable for**GRE** tunnels.

ikey KEY

Specify the input key for the Cisco GRE tunnel. **KEY** must be either in dotted quad (dotted decimal) or a number. Only applicable for**GRE** tunnels.

okey KEY

Same as **ikey**, but set the output key instead. Only applicable for **GRE** tunnels.

icsum Require that all incoming packets have the correct checksum. Only applicable for **GRE** tunnels.

osum Calculate checksums for all outgoing packets. Only applicable for **GRE** tunnels.

ttl TTL

Specify the **Time-To-Live** value for the IP tunnel. **TTL** must be a value between **1** and **255**, or the word **inherit**, which causes the field to be copied from the original IP header.

tos TOS

Specify the **Type-Of-Service** value for the IP tunnel. **T OS** must be a value between **1** and **255**, or the word **inherit**, which causes the field to be copied from the original IP header.

nopmtudisc

Disable **Path MTU Discovery** on this tunnel. Note that a fixed **ttl** is incompatible with this option.

dev PHYS_DEV

Bind the tunnel to the device **PHYS_DEV**.

Homepage

<<https://sourceforge.net/projects/net-tools/>>

AUTHORS

This manpage was written by Sergio Durigan Junior <sergiодj (at) sergiодj (dot) net>. A few parts were inspired by **ip-tunnel(1)**'s manpage.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU General Public License, Version 2 or any later version published by the Free Software Foundation.

NAME

journalctl – Query the systemd journal

SYNOPSIS

journalctl [OPTIONS...] [MATCHES...]

DESCRIPTION

journalctl may be used to query the contents of the **systemd(1)** journal as written by **systemd-journald.service(8)**.

If called without parameters, it will show the full contents of the journal, starting with the oldest entry collected.

If one or more match arguments are passed, the output is filtered accordingly. A match is in the format "**FIELD=VALUE**", e.g. "**_SYSTEMD_UNIT=httpd.service**", referring to the components of a structured journal entry. See **systemd.journal-fields(7)** for a list of well-known fields. If multiple matches are specified matching different fields, the log entries are filtered by both, i.e. the resulting output will show only entries matching all the specified matches of this kind. If two matches apply to the same field, then they are automatically matched as alternatives, i.e. the resulting output will show entries matching any of the specified matches for the same field. Finally, the character "+" may appear as a separate word between other terms on the command line. This causes all matches before and after to be combined in a disjunction (i.e. logical OR).

It is also possible to filter the entries by specifying an absolute file path as an argument. The file path may be a file or a symbolic link and the file must exist at the time of the query. If a file path refers to an executable binary, an "**_EXE=**" match for the canonicalized binary path is added to the query. If a file path refers to an executable script, a "**_COMM=**" match for the script name is added to the query. If a file path refers to a device node, "**_KERNEL_DEVICE=**" matches for the kernel name of the device and for each of its ancestor devices is added to the query. Symbolic links are dereferenced, kernel names are synthesized, and parent devices are identified from the environment at the time of the query. In general, a device node is the best proxy for an actual device, as log entries do not usually contain fields that identify an actual device. For the resulting log entries to be correct for the actual device, the relevant parts of the environment at the time the entry was logged, in particular the actual device corresponding to the device node, must have been the same as those at the time of the query. Because device nodes generally change their corresponding devices across reboots, specifying a device node path causes the resulting entries to be restricted to those from the current boot.

Additional constraints may be added using options **--boot**, **--unit=**, etc., to further limit what entries will be shown (logical AND).

Output is interleaved from all accessible journal files, whether they are rotated or currently being written, and regardless of whether they belong to the system itself or are accessible user journals. The **--header** option can be used to identify which files *are* being shown.

The set of journal files which will be used can be modified using the **--user**, **--system**, **--directory**, and **--file** options, see below.

All users are granted access to their private per-user journals. However, by default, only root and users who are members of a few special groups are granted access to the system journal and the journals of other users. Members of the groups "systemd-journal", "adm", and "wheel" can read all journal files. Note that the two latter groups traditionally have additional privileges specified by the distribution. Members of the "wheel" group can often perform administrative tasks.

The output is paged through **less** by default, and long lines are "truncated" to screen width. The hidden part can be viewed by using the left-arrow and right-arrow keys. Paging can be disabled; see the **--no-pager** option and the "Environment" section below.

When outputting to a tty, lines are colored according to priority: lines of level ERROR and higher are colored red; lines of level NOTICE and higher are highlighted; lines of level DEBUG are colored lighter grey; other lines are displayed normally.

OPTIONS

The following options are understood:

--no-full, --full, -l

Ellipsize fields when they do not fit in available columns. The default is to show full fields, allowing them to wrap or be truncated by the pager, if one is used.

The old options **-l**/**--full** are not useful anymore, except to undo **--no-full**.

-a, --all

Show all fields in full, even if they include unprintable characters or are very long. By default, fields with unprintable characters are abbreviated as "blob data". (Note that the pager may escape unprintable characters again.)

-f, --follow

Show only the most recent journal entries, and continuously print new entries as they are appended to the journal.

-e, --pager-end

Immediately jump to the end of the journal inside the implied pager tool. This implies **-n1000** to guarantee that the pager will not buffer logs of unbounded size. This may be overridden with an explicit **-n** with some other numeric value, while **-null** will disable this cap. Note that this option is only supported for the **less(1)** pager.

--n, --lines=

Show the most recent journal events and limit the number of events shown. If **--follow** is used, this option is implied. The argument is a positive integer or "all" to disable line limiting. The default value is 10 if no argument is given.

--no-tail

Show all stored output lines, even in follow mode. Undoes the effect of **--lines=**.

-r, --reverse

Reverse output so that the newest entries are displayed first.

-o, --output=

Controls the formatting of the journal entries that are shown. Takes one of the following options:

short

is the default and generates an output that is mostly identical to the formatting of classic syslog files, showing one line per journal entry.

short-full

is very similar, but shows timestamps in the format the **--since=** and **--until=** options accept. Unlike the timestamp information shown in **short** output mode this mode includes weekday, year and timezone information in the output, and is locale-independent.

short-iso

is very similar, but shows ISO 8601 wallclock timestamps.

short-iso-precise

as for **short-iso** but includes full microsecond precision.

short-precise

is very similar, but shows classic syslog timestamps with full microsecond precision.

short-monotonic

is very similar, but shows monotonic timestamps instead of wallclock timestamps.

short-unix

is very similar, but shows seconds passed since January 1st 1970 UTC instead of wallclock timestamps ("UNIX time"). The time is shown with microsecond accuracy.

verbose

shows the full-structured entry items with all fields.

export

serializes the journal into a binary (but mostly text-based) stream suitable for backups and network transfer (see [Journal Export Format^{\[1\]}](#) for more information). To import the binary stream back into native journald format use [systemd-journal-remote\(8\)](#).

json

formats entries as JSON objects, separated by newline characters (see [Journal JSON Format^{\[2\]}](#) for more information). Field values are generally encoded as JSON strings, with three exceptions:

1. Fields larger than 4096 bytes are encoded as **null** values. (This may be turned off by passing **--all**, but be aware that this may allocate overly long JSON objects.)
2. Journal entries permit non-unique fields within the same log entry. JSON does not allow non-unique fields within objects. Due to this, if a non-unique field is encountered a JSON array is used as field value, listing all field values as elements.
3. Fields containing non-printable or non-UTF8 bytes are encoded as arrays containing the raw bytes individually formatted as unsigned numbers.

Note that this encoding is reversible (with the exception of the size limit).

json-pretty

formats entries as JSON data structures, but formats them in multiple lines in order to make them more readable by humans.

json-sse

formats entries as JSON data structures, but wraps them in a format suitable for [Server-Sent Events^{\[3\]}](#).

json-seq

formats entries as JSON data structures, but prefixes them with an ASCII Record Separator character (0x1E) and suffixes them with an ASCII Line Feed character (0x0A), in accordance with [JavaScript Object Notation \(JSON\) Text Sequences^{\[4\]}](#) ("application/json-seq").

cat

generates a very terse output, only showing the actual message of each journal entry with no metadata, not even a timestamp. If combined with the **--output-fields=** option will output the listed fields for each log record, instead of the message.

with-unit

similar to short-full, but prefixes the unit and user unit names instead of the traditional syslog identifier. Useful when using templated instances, as it will include the arguments in the unit names.

--output-fields=

A comma separated list of the fields which should be included in the output. This has an effect only for the output modes which would normally show all fields (**verbose**, **export**, **json**, **json-pretty**, **json-sse** and **json-seq**), as well as on **cat**. For the former, the "**__CURSOR**", "**__REALTIME_TIMESTAMP**", "**__MONOTONIC_TIMESTAMP**", and "**_BOOT_ID**" fields are always printed.

--utc

Express time in Coordinated Universal Time (UTC).

--no-hostname

Don't show the hostname field of log messages originating from the local host. This switch has an effect only on the **short** family of output modes (see above).

Note: this option does not remove occurrences of the hostname from log entries themselves, so it does not prevent the hostname from being visible in the logs.

-x, --catalog

Augment log lines with explanation texts from the message catalog. This will add explanatory help texts to log messages in the output where this is available. These short help texts will explain the context of an error or log event, possible solutions, as well as pointers to support forums, developer documentation, and any other relevant manuals. Note that help texts are not available for all messages, but only for selected ones. For more information on the message catalog, please refer to the [Message Catalog Developer Documentation](#)^[5].

Note: when attaching **journalctl** output to bug reports, please do *not* use **-x**.

-q, --quiet

Suppresses all informational messages (i.e. "— Journal begins at ...", "— Reboot —"), any warning messages regarding inaccessible system journals when run as a normal user.

-m, --merge

Show entries interleaved from all available journals, including remote ones.

-b [[ID][±offset]|all], --boot=[ID][±offset]|all

Show messages from a specific boot. This will add a match for "_BOOT_ID=".

The argument may be empty, in which case logs for the current boot will be shown.

If the boot ID is omitted, a positive *offset* will look up the boots starting from the beginning of the journal, and an equal-or-less-than zero *offset* will look up boots starting from the end of the journal. Thus, **1** means the first boot found in the journal in chronological order, **2** the second and so on; while **-0** is the last boot, **-1** the boot before last, and so on. An empty *offset* is equivalent to specifying **-0**, except when the current boot is not the last boot (e.g. because **--directory** was specified to look at logs from a different machine).

If the 32-character *ID* is specified, it may optionally be followed by *offset* which identifies the boot relative to the one given by boot *ID*. Negative values mean earlier boots and positive values mean later boots. If *offset* is not specified, a value of zero is assumed, and the logs for the boot given by *ID* are shown.

The special argument **all** can be used to negate the effect of an earlier use of **-b**.

--list-boots

Show a tabular list of boot numbers (relative to the current boot), their IDs, and the timestamps of the first and last message pertaining to the boot.

-k, --dmesg

Show only kernel messages. This implies **-b** and adds the match "_TRANSPORT=kernel".

-t, --identifier=SYSLOG_IDENTIFIER

Show messages for the specified syslog identifier *SYSLOG_IDENTIFIER*.

This parameter can be specified multiple times.

-u, --unit=UNIT|PATTERN

Show messages for the specified systemd unit *UNIT* (such as a service unit), or for any of the units matched by *PATTERN*. If a pattern is specified, a list of unit names found in the journal is compared with the specified pattern and all that match are used. For each unit name, a match is added for messages from the unit ("_SYSTEMD_UNIT=*UNIT*"), along with additional matches for messages from systemd and messages about core dumps for the specified unit. A match is also added for "_SYSTEMD_SLICE=*UNIT*", such that if the provided *UNIT* is a **systemd.slice(5)** unit, all logs of children of the slice will be shown.

This parameter can be specified multiple times.

--user-unit=

Show messages for the specified user session unit. This will add a match for messages from the unit ("`_SYSTEMD_USER_UNIT=`" and "`_UID=`") and additional matches for messages from session systemd and messages about core dumps for the specified unit. A match is also added for "`_SYSTEMD_USER_SLICE=UNIT`", such that if the provided `UNIT` is a **systemd.slice(5)** unit, all logs of children of the unit will be shown.

This parameter can be specified multiple times.

-p, --priority=

Filter output by message priorities or priority ranges. Takes either a single numeric or textual log level (i.e. between 0/"emerg" and 7/"debug"), or a range of numeric/text log levels in the form FROM..TO. The log levels are the usual syslog log levels as documented in **syslog(3)**, i.e. "emerg" (0), "alert" (1), "crit" (2), "err" (3), "warning" (4), "notice" (5), "info" (6), "debug" (7). If a single log level is specified, all messages with this log level or a lower (hence more important) log level are shown. If a range is specified, all messages within the range are shown, including both the start and the end value of the range. This will add "PRIORITY=" matches for the specified priorities.

--facility=

Filter output by syslog facility. Takes a comma-separated list of numbers or facility names. The names are the usual syslog facilities as documented in **syslog(3)**. **--facility=help** may be used to display a list of known facility names and exit.

-g, --grep=

Filter output to entries where the `MESSAGE=` field matches the specified regular expression. PERL-compatible regular expressions are used, see **pcre2pattern(3)** for a detailed description of the syntax.

If the pattern is all lowercase, matching is case insensitive. Otherwise, matching is case sensitive. This can be overridden with the **--case-sensitive** option, see below.

--case-sensitive[=BOOLEAN]

Make pattern matching case sensitive or case insensitive.

-c, --cursor=

Start showing entries from the location in the journal specified by the passed cursor.

--cursor-file=FILE

If `FILE` exists and contains a cursor, start showing entries *after* this location. Otherwise the show entries according the other given options. At the end, write the cursor of the last entry to `FILE`. Use this option to continually read the journal by sequentially calling **journalctl**.

--after-cursor=

Start showing entries from the location in the journal *after* the location specified by the passed cursor. The cursor is shown when the **--show-cursor** option is used.

--show-cursor

The cursor is shown after the last entry after two dashes:

```
-- cursor: s=0639...
```

The format of the cursor is private and subject to change.

-S, --since=, -U, --until=

Start showing entries on or newer than the specified date, or on or older than the specified date, respectively. Date specifications should be of the format "2012-10-30 18:17:16". If the time part is omitted, "00:00:00" is assumed. If only the seconds component is omitted, ":00" is assumed. If the date component is omitted, the current day is assumed. Alternatively the strings "yesterday", "today", "tomorrow" are understood, which refer to 00:00:00 of the day before the current day, the current day, or the day after the current day, respectively. "now" refers to the current time. Finally, relative times may be specified, prefixed with "-" or "+", referring to times before or after the current time,

respectively. For complete time and date specification, see **systemd.time(7)**. Note that **--output=short-full** prints timestamps that follow precisely this format.

-F, --field=

Print all possible data values the specified field can take in all entries of the journal.

-N, --fields

Print all field names currently used in all entries of the journal.

--system, --user

Show messages from system services and the kernel (with **--system**). Show messages from service of current user (with **--user**). If neither is specified, show all messages that the user can see.

-M, --machine=

Show messages from a running, local container. Specify a container name to connect to.

-D DIR, --directory=DIR

Takes a directory path as argument. If specified, journalctl will operate on the specified journal directory *DIR* instead of the default runtime and system journal paths.

--file=GLOB

Takes a file glob as an argument. If specified, journalctl will operate on the specified journal files matching *GLOB* instead of the default runtime and system journal paths. May be specified multiple times, in which case files will be suitably interleaved.

--root=ROOT

Takes a directory path as an argument. If specified, **journalctl** will operate on journal directories and catalog file hierarchy underneath the specified directory instead of the root directory (e.g.

--update-catalog will create *ROOT*/var/lib/systemd/catalog/database, and journal files under *ROOT*/run/journal/ or *ROOT*/var/log/journal/ will be displayed).

--image=IMAGE

Takes a path to a disk image file or block device node. If specified, **journalctl** will operate on the file system in the indicated disk image. This is similar to **--root=** but operates on file systems stored in disk images or block devices, thus providing an easy way to extract log data from disk images. The disk image should either contain just a file system or a set of file systems within a GPT partition table, following the [Discoverable Partitions Specification](#)^[6]. For further information on supported disk images, see **systemd-nspawn(1)**'s switch of the same name.

--namespace=NAMESPACE

Takes a journal namespace identifier string as argument. If not specified the data collected by the default namespace is shown. If specified shows the log data of the specified namespace instead. If the namespace is specified as "*" data from all namespaces is shown, interleaved. If the namespace identifier is prefixed with "+" data from the specified namespace and the default namespace is shown, interleaved, but no other. For details about journal namespaces see **systemd-journald.service(8)**.

--header

Instead of showing journal contents, show internal header information of the journal fields accessed.

This option is particularly useful when trying to identify out-of-order journal entries, as happens for example when the machine is booted with the wrong system time.

--disk-usage

Shows the current disk usage of all journal files. This shows the sum of the disk usage of all archived and active journal files.

--vacuum-size=, --vacuum-time=, --vacuum-files=

Removes the oldest archived journal files until the disk space they use falls below the specified size (specified with the usual "K", "M", "G" and "T" suffixes), or all archived journal files contain no data older than the specified timespan (specified with the usual "s", "m", "h", "days", "months", "weeks" and "years" suffixes), or no more than the specified number of separate journal files remain. Note that running **--vacuum-size=** has only an indirect effect on the output shown by **--disk-usage**, as the

latter includes active journal files, while the vacuuming operation only operates on archived journal files. Similarly, **--vacuum-files=** might not actually reduce the number of journal files to below the specified number, as it will not remove active journal files.

--vacuum-size=, **--vacuum-time=** and **--vacuum-files=** may be combined in a single invocation to enforce any combination of a size, a time and a number of files limit on the archived journal files. Specifying any of these three parameters as zero is equivalent to not enforcing the specific limit, and is thus redundant.

These three switches may also be combined with **--rotate** into one command. If so, all active files are rotated first, and the requested vacuuming operation is executed right after. The rotation has the effect that all currently active files are archived (and potentially new, empty journal files opened as replacement), and hence the vacuuming operation has the greatest effect as it can take all log data written so far into account.

--list-catalog [128-bit-ID...]

List the contents of the message catalog as a table of message IDs, plus their short description strings.

If any *128-bit-IDs* are specified, only those entries are shown.

--dump-catalog [128-bit-ID...]

Show the contents of the message catalog, with entries separated by a line consisting of two dashes and the ID (the format is the same as .catalog files).

If any *128-bit-IDs* are specified, only those entries are shown.

--update-catalog

Update the message catalog index. This command needs to be executed each time new catalog files are installed, removed, or updated to rebuild the binary catalog index.

--setup-keys

Instead of showing journal contents, generate a new key pair for Forward Secure Sealing (FSS). This will generate a sealing key and a verification key. The sealing key is stored in the journal data directory and shall remain on the host. The verification key should be stored externally. Refer to the **Seal=** option in **journald.conf(5)** for information on Forward Secure Sealing and for a link to a refereed scholarly paper detailing the cryptographic theory it is based on.

--force

When **--setup-keys** is passed and Forward Secure Sealing (FSS) has already been configured, recreate FSS keys.

--interval=

Specifies the change interval for the sealing key when generating an FSS key pair with **--setup-keys**. Shorter intervals increase CPU consumption but shorten the time range of undetectable journal alterations. Defaults to 15min.

--verify

Check the journal file for internal consistency. If the file has been generated with FSS enabled and the FSS verification key has been specified with **--verify-key=**, authenticity of the journal file is verified.

--verify-key=

Specifies the FSS verification key to use for the **--verify** operation.

--sync

Asks the journal daemon to write all yet unwritten journal data to the backing file system and synchronize all journals. This call does not return until the synchronization operation is complete. This command guarantees that any log messages written before its invocation are safely stored on disk at the time it returns.

--flush

Asks the journal daemon to flush any log data stored in `/run/log/journal/` into `/var/log/journal/`, if persistent storage is enabled. This call does not return until the operation is complete. Note that this call is idempotent: the data is only flushed from `/run/log/journal/` into `/var/log/journal/` once during system runtime (but see **--relinquish-var** below), and this command exits cleanly without executing any operation if this has already happened. This command effectively guarantees that all data is flushed to `/var/log/journal/` at the time it returns.

--relinquish-var

Asks the journal daemon for the reverse operation to **--flush**: if requested the daemon will write further log data to `/run/log/journal/` and stops writing to `/var/log/journal/`. A subsequent call to **--flush** causes the log output to switch back to `/var/log/journal/`, see above.

--smart-relinquish-var

Similar to **--relinquish-var** but executes no operation if the root file system and `/var/lib/journal/` reside on the same mount point. This operation is used during system shutdown in order to make the journal daemon stop writing data to `/var/log/journal/` in case that directory is located on a mount point that needs to be unmounted.

--rotate

Asks the journal daemon to rotate journal files. This call does not return until the rotation operation is complete. Journal file rotation has the effect that all currently active journal files are marked as archived and renamed, so that they are never written to in future. New (empty) journal files are then created in their place. This operation may be combined with **--vacuum-size=**, **--vacuum-time=** and **--vacuum-file=** into a single command, see above.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--no-pager

Do not pipe output into a pager.

EXIT STATUS

On success, 0 is returned; otherwise, a non-zero failure code is returned.

ENVIRONMENT

\$SYSTEMD_LOG_LEVEL

The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog(3)** for more information.

\$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This setting is only useful when messages are written directly to the terminal, because **journalctl(1)** and other tools that display logs will color messages based on the log level on their own.

\$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This setting is only useful when messages are written directly to the terminal or a file, because **journalctl(1)** and other tools that display logs will attach timestamps based on the entry metadata on their own.

\$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

Note that the log location is often attached as metadata to journal entries anyway. Including it directly

in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TID

A boolean. If true, messages will be prefixed with the current numerical thread ID (TID).

Note that the this information is attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console-prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see [syslog\(3\)](#)), **kmsg** (log to the kernel circular log buffer), **journal** (log to the journal), **journal-or-kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

\$SYSTEMD_PAGER

Pager to use when **--no-pager** is not given; overrides **\$PAGER**. If neither **\$SYSTEMD_PAGER** nor **\$PAGER** are set, a set of well-known pager implementations are tried in turn, including **less(1)** and **more(1)**, until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **--no-pager**.

\$SYSTEMD_LESS

Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K

This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of **\$SYSTEMD_LESS** does not include "K", and the pager that is invoked is **less**, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X

This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See **less(1)** for more discussion.

\$SYSTEMD_LESSCHARSET

Override the charset passed to **less** (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

\$SYSTEMD_PAGERSECURE

Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If **\$SYSTEMD_PAGERSECURE** is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see **geteuid(2)** and **sd_pid_get_owner_uid(3)**. In secure mode, **LESSSECURE=1** will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When **\$SYSTEMD_PAGERSECURE** is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only **less(1)** implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under **sudo(8)** or **pkexec(1)**, care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as described above. Setting **SYSTEMD_PAGERSECURE=0** or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the **\$SYSTEMD_PAGER** or **\$PAGER** variables are to be honoured,

\$SYSTEMD_PAGERSECURE must be set too. It might be reasonable to completely disable the pager using **--no-pager** instead.

\$SYSTEMD_COLORS

Takes a boolean argument. When true, **systemd** and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on **\$TERM** and what the console is connected to.

\$SYSTEMD_URLIFY

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that **systemd** makes based on **\$TERM** and other conditions.

EXAMPLES

Without arguments, all collected logs are shown unfiltered:

```
journalctl
```

With one match specified, all entries with a field matching the expression are shown:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service
```

```
journalctl _SYSTEMD_CGROUP=/user.slice/user-42.slice/session-c1.scope
```

If two different fields are matched, only entries matching both expressions at the same time are shown:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service _PID=28097
```

If two matches refer to the same field, all entries matching either expression are shown:

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service _SYSTEMD_UNIT=dbus.service
```

If the separator "+" is used, two expressions may be combined in a logical OR. The following will show all messages from the Avahi service process with the PID 28097 plus all messages from the D-Bus service (from any of its processes):

```
journalctl _SYSTEMD_UNIT=avahi-daemon.service _PID=28097 + _SYSTEMD_UNIT=dbus.service
```

To show all fields emitted *by* a unit and *about* the unit, option **-u**/**--unit=** should be used. **journalctl -u name** expands to a complex filter similar to

```
_SYSTEMD_UNIT=name.service
+ UNIT=name.service _PID=1
+ OBJECT_SYSTEMD_UNIT=name.service _UID=0
+ COREDUMP_UNIT=name.service _UID=0 MESSAGE_ID=fc2e22bc6ee647b6b90729ab34a250b1
```

(see **systemd.journal-fields(7)** for an explanation of those patterns).

Show all logs generated by the D-Bus executable:

```
journalctl /usr/bin/dbus-daemon
```

Show all kernel logs from previous boot:

```
journalctl -k -b -1
```

Show a live log display from a system service apache.service:

```
journalctl -f -u apache
```

SEE ALSO

systemd(1), systemd-journald.service(8), systemctl(1), coredumpctl(1), systemd.journal-fields(7), journald.conf(5), systemd.time(7), systemd-journal-remote.service(8), systemd-journal-upload.service(8)

NOTES

1. Journal Export Format
https://systemd.io/JOURNAL_EXPORT_FORMATS#journal-export-format
2. Journal JSON Format
https://systemd.io/JOURNAL_EXPORT_FORMATS#journal-json-format
3. Server-Sent Events
https://developer.mozilla.org/en-US/docs/Server-sent_events/Using_server-sent_events
4. JavaScript Object Notation (JSON) Text Sequences
<https://tools.ietf.org/html/rfc7464>
5. Message Catalog Developer Documentation
<https://www.freedesktop.org/wiki/Software/systemd/catalog>
6. Discoverable Partitions Specification
https://systemd.io/DISCOVERABLE_PARTITIONS

NAME

journald.conf, journald.conf.d, journald@.conf – Journal service configuration files

SYNOPSIS

```
/etc/systemd/journald.conf
/etc/systemd/journald.conf.d/*.conf
/run/systemd/journald.conf.d/*.conf
/usr/lib/systemd/journald.conf.d/*.conf
/etc/systemd/journald@NAMESPACE.conf
/etc/systemd/journald@NAMESPACE.conf.d/*.conf
/run/systemd/journald@NAMESPACE.conf.d/*.conf
/usr/lib/systemd/journald@NAMESPACE.conf.d/*.conf
```

DESCRIPTION

These files configure various parameters of the systemd journal service, **systemd-journald.service(8)**. See **systemd.syntax(7)** for a general description of the syntax.

The **systemd-journald** instance managing the default namespace is configured by `/etc/systemd/journald.conf` and associated drop-ins. Instances managing other namespaces read `/etc/systemd/journald@NAMESPACE.conf` and associated drop-ins with the namespace identifier filled in. This allows each namespace to carry a distinct configuration. See **systemd-journald.service(8)** for details about journal namespaces.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in `/etc/systemd/` contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from `/usr/lib/systemd/*.conf.d/`, `/usr/local/lib/systemd/*.conf.d/`, and `/etc/systemd/*.conf.d/`. Those drop-ins have higher precedence and override the main configuration file. Files in the `*.conf.d/` configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under `/usr/`. Files in `/etc/` are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to `/dev/null` in the configuration directory in `/etc/`, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the [Journal] section:

Storage=

Controls where to store journal data. One of "volatile", "persistent", "auto" and "none". If "volatile", journal log data will be stored only in memory, i.e. below the `/run/log/journal` hierarchy (which is created if needed). If "persistent", data will be stored preferably on disk, i.e. below the `/var/log/journal` hierarchy (which is created if needed), with a fallback to `/run/log/journal` (which is created if needed), during early boot and if the disk is not writable. "auto" behaves like "persistent" if the `/var/log/journal` directory exists, and "volatile" otherwise (the existence of the directory controls the storage mode).

"none" turns off all storage, all log data received will be dropped (but forwarding to other targets, such as the console, the kernel log buffer, or a syslog socket will still work). Defaults to "auto" in the default journal namespace, and "persistent" in all others.

Note that journald will initially use volatile storage, until a call to **journalctl --flush** (or sending **SIGUSR1** to journald) will cause it to switch to persistent logging (under the conditions mentioned above). This is done automatically on boot via "systemd-journal-flush.service".

Note that when this option is changed to "volatile", existing persistent data is not removed. In the other direction, **journalctl(1)** with the **--flush** option may be used to move volatile data to persistent storage.

Compress=

Can take a boolean value. If enabled (the default), data objects that shall be stored in the journal and are larger than the default threshold of 512 bytes are compressed before they are written to the file system. It can also be set to a number of bytes to specify the compression threshold directly. Suffixes like K, M, and G can be used to specify larger units.

Seal=

Takes a boolean value. If enabled (the default), and a sealing key is available (as created by **journalctl(1)**'s **--setup-keys** command), Forward Secure Sealing (FSS) for all persistent journal files is enabled. FSS is based on [Seekable Sequential Key Generators^{\[1\]}](#) by G. A. Marson and B. Poettering (doi:10.1007/978-3-642-40203-6_7) and may be used to protect journal files from unnoticed alteration.

SplitMode=

Controls whether to split up journal files per user, either "uid" or "none". Split journal files are primarily useful for access control: on UNIX/Linux access control is managed per file, and the journal daemon will assign users read access to their journal files. If "uid", all regular users (with UID outside the range of system users, dynamic service users, and the nobody user) will each get their own journal files, and system users will log to the system journal. See [Users, Groups, UIDs and GIDs on systemd systems^{\[2\]}](#) for more details about UID ranges. If "none", journal files are not split up by user and all messages are instead stored in the single system journal. In this mode unprivileged users generally do not have access to their own log data. Note that splitting up journal files by user is only available for journals stored persistently. If journals are stored on volatile storage (see *Storage=* above), only a single journal file is used. Defaults to "uid".

RateLimitIntervalSec=, RateLimitBurst=

Configures the rate limiting that is applied to all messages generated on the system. If, in the time interval defined by *RateLimitIntervalSec=*, more messages than specified in *RateLimitBurst=* are logged by a service, all further messages within the interval are dropped until the interval is over. A message about the number of dropped messages is generated. This rate limiting is applied per-service, so that two services which log do not interfere with each other's limits. Defaults to 10000 messages in 30s. The time specification for *RateLimitIntervalSec=* may be specified in the following units: "s", "min", "h", "ms", "us". To turn off any kind of rate limiting, set either value to 0.

Note that the effective rate limit is multiplied by a factor derived from the available free disk space for the journal. Currently, this factor is calculated using the base 2 logarithm.

Table 1. Example *RateLimitBurst=* rate modifications by the available disk space

Available Disk Space	Burst Multiplier
<= 1MB	1
<= 16MB	2
<= 256MB	3
<= 4GB	4
<= 64GB	5
<= 1TB	6

If a service provides rate limits for itself through *LogRateLimitIntervalSec*= and/or *LogRateLimitBurst*= in **systemd.exec**(5), those values will override the settings specified here.

SystemMaxUse=, *SystemKeepFree*=, *SystemMaxFileSize*=, *SystemMaxFiles*=, *RuntimeMaxUse*=, *RuntimeKeepFree*=, *RuntimeMaxFileSize*=, *RuntimeMaxFiles*=

Enforce size limits on the journal files stored. The options prefixed with "System" apply to the journal files when stored on a persistent file system, more specifically /var/log/journal. The options prefixed with "Runtime" apply to the journal files when stored on a volatile in-memory file system, more specifically /run/log/journal. The former is used only when /var/ is mounted, writable, and the directory /var/log/journal exists. Otherwise, only the latter applies. Note that this means that during early boot and if the administrator disabled persistent logging, only the latter options apply, while the former apply if persistent logging is enabled and the system is fully booted up. **journaldctl** and **systemd-journald** ignore all files with names not ending with ".journal" or ".journal~", so only such files, located in the appropriate directories, are taken into account when calculating current disk usage.

SystemMaxUse= and *RuntimeMaxUse*= control how much disk space the journal may use up at most. *SystemKeepFree*= and *RuntimeKeepFree*= control how much disk space **systemd-journald** shall leave free for other uses. **systemd-journald** will respect both limits and use the smaller of the two values.

The first pair defaults to 10% and the second to 15% of the size of the respective file system, but each value is capped to 4G. If the file system is nearly full and either *SystemKeepFree*= or *RuntimeKeepFree*= are violated when **systemd-journald** is started, the limit will be raised to the percentage that is actually free. This means that if there was enough free space before and journal files were created, and subsequently something else causes the file system to fill up, **journald** will stop using more space, but it will not be removing existing files to reduce the footprint again, either. Also note that only archived files are deleted to reduce the space occupied by journal files. This means that, in effect, there might still be more space used than *SystemMaxUse*= or *RuntimeMaxUse*= limit after a vacuuming operation is complete.

SystemMaxFileSize= and *RuntimeMaxFileSize*= control how large individual journal files may grow at most. This influences the granularity in which disk space is made available through rotation, i.e. deletion of historic data. Defaults to one eighth of the values configured with *SystemMaxUse*= and *RuntimeMaxUse*=, so that usually seven rotated journal files are kept as history.

Specify values in bytes or use K, M, G, T, P, E as units for the specified sizes (equal to 1024, 1024², ... bytes). Note that size limits are enforced synchronously when journal files are extended, and no explicit rotation step triggered by time is needed.

SystemMaxFiles= and *RuntimeMaxFiles*= control how many individual journal files to keep at most. Note that only archived files are deleted to reduce the number of files until this limit is reached; active files will stay around. This means that, in effect, there might still be more journal files around in total than this limit after a vacuuming operation is complete. This setting defaults to 100.

MaxFileSec=

The maximum time to store entries in a single journal file before rotating to the next one. Normally, time-based rotation should not be required as size-based rotation with options such as *SystemMaxFileSize*= should be sufficient to ensure that journal files do not grow without bounds.

However, to ensure that not too much data is lost at once when old journal files are deleted, it might make sense to change this value from the default of one month. Set to 0 to turn off this feature. This setting takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

MaxRetentionSec=

The maximum time to store journal entries. This controls whether journal files containing entries older than the specified time span are deleted. Normally, time-based deletion of old journal files should not be required as size-based deletion with options such as *SystemMaxUse=* should be sufficient to ensure that journal files do not grow without bounds. However, to enforce data retention policies, it might make sense to change this value from the default of 0 (which turns off this feature). This setting also takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

SyncIntervalSec=

The timeout before synchronizing journal files to disk. After syncing, journal files are placed in the OFFLINE state. Note that syncing is unconditionally done immediately after a log message of priority CRIT, ALERT or EMERG has been logged. This setting hence applies only to messages of the levels ERR, WARNING, NOTICE, INFO, DEBUG. The default timeout is 5 minutes.

ForwardToSyslog=, *ForwardToKMsg=*, *ForwardToConsole=*, *ForwardToWall=*

Control whether log messages received by the journal daemon shall be forwarded to a traditional syslog daemon, to the kernel log buffer (kmsg), to the system console, or sent as wall messages to all logged-in users. These options take boolean arguments. If forwarding to syslog is enabled but nothing reads messages from the socket, forwarding to syslog has no effect. By default, only forwarding to syslog and wall is enabled. These settings may be overridden at boot time with the kernel command line options "systemd.journald.forward_to_syslog", "systemd.journald.forward_to_kmsg", "systemd.journald.forward_to_console", and "systemd.journald.forward_to_wall". If the option name is specified without "=" and the following argument, true is assumed. Otherwise, the argument is parsed as a boolean.

When forwarding to the console, the TTY to log to can be changed with *TTYPath=*, described below.

When forwarding to the kernel log buffer (kmsg), make sure to select a suitably large size for the log buffer, for example by adding "log_buf_len=8M" to the kernel command line. **systemd** will automatically disable kernel's rate-limiting applied to userspace processes (equivalent to setting "printk.devkmsg=on").

MaxLevelStore=, *MaxLevelSyslog=*, *MaxLevelKMsg=*, *MaxLevelConsole=*, *MaxLevelWall=*

Controls the maximum log level of messages that are stored in the journal, forwarded to syslog, kmsg, the console or wall (if that is enabled, see above). As argument, takes one of "emerg", "alert", "crit", "err", "warning", "notice", "info", "debug", or integer values in the range of 0–7 (corresponding to the same levels). Messages equal or below the log level specified are stored/forwarded, messages above are dropped. Defaults to "debug" for *MaxLevelStore=* and *MaxLevelSyslog=*, to ensure that all messages are stored in the journal and forwarded to syslog. Defaults to "notice" for *MaxLevelKMsg=*, "info" for *MaxLevelConsole=*, and "emerg" for *MaxLevelWall=*. These settings may be overridden at boot time with the kernel command line options "systemd.journald.max_level_store=", "systemd.journald.max_level_syslog=", "systemd.journald.max_level_kmsg=", "systemd.journald.max_level_console=", "systemd.journald.max_level_wall=".

ReadKMsg=

Takes a boolean value. If enabled **systemd-journal** processes /dev/kmsg messages generated by the kernel. In the default journal namespace this option is enabled by default, it is disabled in all others.

Audit=

Takes a boolean value. If enabled **systemd-journal** will turn on kernel auditing on start-up. If disabled it will turn it off. If unset it will neither enable nor disable it, leaving the previous state unchanged. Note that this option does not control whether **systemd-journald** collects generated audit

records, it just controls whether it tells the kernel to generate them. This means if another tool turns on auditing even if **systemd-journald** left it off, it will still collect the generated messages. Defaults to off.

TTYPath=

Change the console TTY to use if *ForwardToConsole=yes* is used. Defaults to /dev/console.

LineMax=

The maximum line length to permit when converting stream logs into record logs. When a systemd unit's standard output/error are connected to the journal via a stream socket, the data read is split into individual log records at newline ("\\n", ASCII 10) and **NUL** characters. If no such delimiter is read for the specified number of bytes a hard log record boundary is artificially inserted, breaking up overly long lines into multiple log records. Selecting overly large values increases the possible memory usage of the Journal daemon for each stream client, as in the worst case the journal daemon needs to buffer the specified number of bytes in memory before it can flush a new log record to disk. Also note that permitting overly large line maximum line lengths affects compatibility with traditional log protocols as log records might not fit anymore into a single **AF_UNIX** or **AF_INET** datagram. Takes a size in bytes. If the value is suffixed with K, M, G or T, the specified size is parsed as Kilobytes, Megabytes, Gigabytes, or Terabytes (with the base 1024), respectively. Defaults to 48K, which is relatively large but still small enough so that log records likely fit into network datagrams along with extra room for metadata. Note that values below 79 are not accepted and will be bumped to 79.

FORWARDING TO TRADITIONAL SYSLOG DAEMONS

Journal events can be transferred to a different logging daemon in two different ways. With the first method, messages are immediately forwarded to a socket (/run/systemd/journal/syslog), where the traditional syslog daemon can read them. This method is controlled by the *ForwardToSyslog=* option. With a second method, a syslog daemon behaves like a normal journal client, and reads messages from the journal files, similarly to **journalctl(1)**. With this, messages do not have to be read immediately, which allows a logging daemon which is only started late in boot to access all messages since the start of the system. In addition, full structured meta-data is available to it. This method of course is available only if the messages are stored in a journal file at all. So it will not work if *Storage=none* is set. It should be noted that usually the *second* method is used by syslog daemons, so the *Storage=* option, and not the *ForwardToSyslog=* option, is relevant for them.

SEE ALSO

systemd(1), **systemd-journald.service(8)**, **journalctl(1)**, **systemd.journal-fields(7)**, **systemd-system.conf(5)**

NOTES

1. Seekable Sequential Key Generators
<https://eprint.iacr.org/2013/397>
2. Users, Groups, UIDs and GIDs on systemd systems
<https://systemd.io/UIDS-GIDS>

NAME

journald.conf, journald.conf.d, journald@.conf – Journal service configuration files

SYNOPSIS

```
/etc/systemd/journald.conf
/etc/systemd/journald.conf.d/*.conf
/run/systemd/journald.conf.d/*.conf
/usr/lib/systemd/journald.conf.d/*.conf
/etc/systemd/journald@NAMESPACE.conf
/etc/systemd/journald@NAMESPACE.conf.d/*.conf
/run/systemd/journald@NAMESPACE.conf.d/*.conf
/usr/lib/systemd/journald@NAMESPACE.conf.d/*.conf
```

DESCRIPTION

These files configure various parameters of the systemd journal service, **systemd-journald.service(8)**. See **systemd.syntax(7)** for a general description of the syntax.

The **systemd-journald** instance managing the default namespace is configured by `/etc/systemd/journald.conf` and associated drop-ins. Instances managing other namespaces read `/etc/systemd/journald@NAMESPACE.conf` and associated drop-ins with the namespace identifier filled in. This allows each namespace to carry a distinct configuration. See **systemd-journald.service(8)** for details about journal namespaces.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in `/etc/systemd/` contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from `/usr/lib/systemd/*.conf.d/`, `/usr/local/lib/systemd/*.conf.d/`, and `/etc/systemd/*.conf.d/`. Those drop-ins have higher precedence and override the main configuration file. Files in the `*.conf.d/` configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under `/usr/`. Files in `/etc/` are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to `/dev/null` in the configuration directory in `/etc/`, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the [Journal] section:

Storage=

Controls where to store journal data. One of "volatile", "persistent", "auto" and "none". If "volatile", journal log data will be stored only in memory, i.e. below the `/run/log/journal` hierarchy (which is created if needed). If "persistent", data will be stored preferably on disk, i.e. below the `/var/log/journal` hierarchy (which is created if needed), with a fallback to `/run/log/journal` (which is created if needed), during early boot and if the disk is not writable. "auto" behaves like "persistent" if the `/var/log/journal` directory exists, and "volatile" otherwise (the existence of the directory controls the storage mode).

"none" turns off all storage, all log data received will be dropped (but forwarding to other targets, such as the console, the kernel log buffer, or a syslog socket will still work). Defaults to "auto" in the default journal namespace, and "persistent" in all others.

Note that journald will initially use volatile storage, until a call to **journalctl --flush** (or sending **SIGUSR1** to journald) will cause it to switch to persistent logging (under the conditions mentioned above). This is done automatically on boot via "systemd-journal-flush.service".

Note that when this option is changed to "volatile", existing persistent data is not removed. In the other direction, **journalctl(1)** with the **--flush** option may be used to move volatile data to persistent storage.

Compress=

Can take a boolean value. If enabled (the default), data objects that shall be stored in the journal and are larger than the default threshold of 512 bytes are compressed before they are written to the file system. It can also be set to a number of bytes to specify the compression threshold directly. Suffixes like K, M, and G can be used to specify larger units.

Seal=

Takes a boolean value. If enabled (the default), and a sealing key is available (as created by **journalctl(1)**'s **--setup-keys** command), Forward Secure Sealing (FSS) for all persistent journal files is enabled. FSS is based on [Seekable Sequential Key Generators^{\[1\]}](#) by G. A. Marson and B. Poettering (doi:10.1007/978-3-642-40203-6_7) and may be used to protect journal files from unnoticed alteration.

SplitMode=

Controls whether to split up journal files per user, either "uid" or "none". Split journal files are primarily useful for access control: on UNIX/Linux access control is managed per file, and the journal daemon will assign users read access to their journal files. If "uid", all regular users (with UID outside the range of system users, dynamic service users, and the nobody user) will each get their own journal files, and system users will log to the system journal. See [Users, Groups, UIDs and GIDs on systemd systems^{\[2\]}](#) for more details about UID ranges. If "none", journal files are not split up by user and all messages are instead stored in the single system journal. In this mode unprivileged users generally do not have access to their own log data. Note that splitting up journal files by user is only available for journals stored persistently. If journals are stored on volatile storage (see *Storage=* above), only a single journal file is used. Defaults to "uid".

RateLimitIntervalSec=, RateLimitBurst=

Configures the rate limiting that is applied to all messages generated on the system. If, in the time interval defined by *RateLimitIntervalSec=*, more messages than specified in *RateLimitBurst=* are logged by a service, all further messages within the interval are dropped until the interval is over. A message about the number of dropped messages is generated. This rate limiting is applied per-service, so that two services which log do not interfere with each other's limits. Defaults to 10000 messages in 30s. The time specification for *RateLimitIntervalSec=* may be specified in the following units: "s", "min", "h", "ms", "us". To turn off any kind of rate limiting, set either value to 0.

Note that the effective rate limit is multiplied by a factor derived from the available free disk space for the journal. Currently, this factor is calculated using the base 2 logarithm.

Table 1. Example *RateLimitBurst=* rate modifications by the available disk space

Available Disk Space	Burst Multiplier
<= 1MB	1
<= 16MB	2
<= 256MB	3
<= 4GB	4
<= 64GB	5
<= 1TB	6

If a service provides rate limits for itself through *LogRateLimitIntervalSec*= and/or *LogRateLimitBurst*= in **systemd.exec(5)**, those values will override the settings specified here.

SystemMaxUse=, *SystemKeepFree*=, *SystemMaxFileSize*=, *SystemMaxFiles*=, *RuntimeMaxUse*=, *RuntimeKeepFree*=, *RuntimeMaxFileSize*=, *RuntimeMaxFiles*=

Enforce size limits on the journal files stored. The options prefixed with "System" apply to the journal files when stored on a persistent file system, more specifically /var/log/journal. The options prefixed with "Runtime" apply to the journal files when stored on a volatile in-memory file system, more specifically /run/log/journal. The former is used only when /var/ is mounted, writable, and the directory /var/log/journal exists. Otherwise, only the latter applies. Note that this means that during early boot and if the administrator disabled persistent logging, only the latter options apply, while the former apply if persistent logging is enabled and the system is fully booted up. **journalctl** and **systemd-journald** ignore all files with names not ending with ".journal" or ".journal~", so only such files, located in the appropriate directories, are taken into account when calculating current disk usage.

SystemMaxUse= and *RuntimeMaxUse*= control how much disk space the journal may use up at most. *SystemKeepFree*= and *RuntimeKeepFree*= control how much disk space **systemd-journald** shall leave free for other uses. **systemd-journald** will respect both limits and use the smaller of the two values.

The first pair defaults to 10% and the second to 15% of the size of the respective file system, but each value is capped to 4G. If the file system is nearly full and either *SystemKeepFree*= or *RuntimeKeepFree*= are violated when **systemd-journald** is started, the limit will be raised to the percentage that is actually free. This means that if there was enough free space before and journal files were created, and subsequently something else causes the file system to fill up, **journald** will stop using more space, but it will not be removing existing files to reduce the footprint again, either. Also note that only archived files are deleted to reduce the space occupied by journal files. This means that, in effect, there might still be more space used than *SystemMaxUse*= or *RuntimeMaxUse*= limit after a vacuuming operation is complete.

SystemMaxFileSize= and *RuntimeMaxFileSize*= control how large individual journal files may grow at most. This influences the granularity in which disk space is made available through rotation, i.e. deletion of historic data. Defaults to one eighth of the values configured with *SystemMaxUse*= and *RuntimeMaxUse*=, so that usually seven rotated journal files are kept as history.

Specify values in bytes or use K, M, G, T, P, E as units for the specified sizes (equal to 1024, 1024², ... bytes). Note that size limits are enforced synchronously when journal files are extended, and no explicit rotation step triggered by time is needed.

SystemMaxFiles= and *RuntimeMaxFiles*= control how many individual journal files to keep at most. Note that only archived files are deleted to reduce the number of files until this limit is reached; active files will stay around. This means that, in effect, there might still be more journal files around in total than this limit after a vacuuming operation is complete. This setting defaults to 100.

MaxFileSec=

The maximum time to store entries in a single journal file before rotating to the next one. Normally, time-based rotation should not be required as size-based rotation with options such as *SystemMaxFileSize*= should be sufficient to ensure that journal files do not grow without bounds.

However, to ensure that not too much data is lost at once when old journal files are deleted, it might make sense to change this value from the default of one month. Set to 0 to turn off this feature. This setting takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

MaxRetentionSec=

The maximum time to store journal entries. This controls whether journal files containing entries older than the specified time span are deleted. Normally, time-based deletion of old journal files should not be required as size-based deletion with options such as *SystemMaxUse=* should be sufficient to ensure that journal files do not grow without bounds. However, to enforce data retention policies, it might make sense to change this value from the default of 0 (which turns off this feature). This setting also takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

SyncIntervalSec=

The timeout before synchronizing journal files to disk. After syncing, journal files are placed in the OFFLINE state. Note that syncing is unconditionally done immediately after a log message of priority CRIT, ALERT or EMERG has been logged. This setting hence applies only to messages of the levels ERR, WARNING, NOTICE, INFO, DEBUG. The default timeout is 5 minutes.

ForwardToSyslog=, *ForwardToKMsg=*, *ForwardToConsole=*, *ForwardToWall=*

Control whether log messages received by the journal daemon shall be forwarded to a traditional syslog daemon, to the kernel log buffer (kmsg), to the system console, or sent as wall messages to all logged-in users. These options take boolean arguments. If forwarding to syslog is enabled but nothing reads messages from the socket, forwarding to syslog has no effect. By default, only forwarding to syslog and wall is enabled. These settings may be overridden at boot time with the kernel command line options "systemd.journald.forward_to_syslog", "systemd.journald.forward_to_kmsg", "systemd.journald.forward_to_console", and "systemd.journald.forward_to_wall". If the option name is specified without "=" and the following argument, true is assumed. Otherwise, the argument is parsed as a boolean.

When forwarding to the console, the TTY to log to can be changed with *TTYPath=*, described below.

When forwarding to the kernel log buffer (kmsg), make sure to select a suitably large size for the log buffer, for example by adding "log_buf_len=8M" to the kernel command line. **systemd** will automatically disable kernel's rate-limiting applied to userspace processes (equivalent to setting "printk.devkmsg=on").

MaxLevelStore=, *MaxLevelSyslog=*, *MaxLevelKMsg=*, *MaxLevelConsole=*, *MaxLevelWall=*

Controls the maximum log level of messages that are stored in the journal, forwarded to syslog, kmsg, the console or wall (if that is enabled, see above). As argument, takes one of "emerg", "alert", "crit", "err", "warning", "notice", "info", "debug", or integer values in the range of 0–7 (corresponding to the same levels). Messages equal or below the log level specified are stored/forwarded, messages above are dropped. Defaults to "debug" for *MaxLevelStore=* and *MaxLevelSyslog=*, to ensure that all messages are stored in the journal and forwarded to syslog. Defaults to "notice" for *MaxLevelKMsg=*, "info" for *MaxLevelConsole=*, and "emerg" for *MaxLevelWall=*. These settings may be overridden at boot time with the kernel command line options "systemd.journald.max_level_store=", "systemd.journald.max_level_syslog=", "systemd.journald.max_level_kmsg=", "systemd.journald.max_level_console=", "systemd.journald.max_level_wall=".

ReadKMsg=

Takes a boolean value. If enabled **systemd-journal** processes /dev/kmsg messages generated by the kernel. In the default journal namespace this option is enabled by default, it is disabled in all others.

Audit=

Takes a boolean value. If enabled **systemd-journal** will turn on kernel auditing on start-up. If disabled it will turn it off. If unset it will neither enable nor disable it, leaving the previous state unchanged. Note that this option does not control whether **systemd-journald** collects generated audit

records, it just controls whether it tells the kernel to generate them. This means if another tool turns on auditing even if **systemd-journald** left it off, it will still collect the generated messages. Defaults to off.

TTYPath=

Change the console TTY to use if *ForwardToConsole=yes* is used. Defaults to /dev/console.

LineMax=

The maximum line length to permit when converting stream logs into record logs. When a systemd unit's standard output/error are connected to the journal via a stream socket, the data read is split into individual log records at newline ("\n", ASCII 10) and **NUL** characters. If no such delimiter is read for the specified number of bytes a hard log record boundary is artificially inserted, breaking up overly long lines into multiple log records. Selecting overly large values increases the possible memory usage of the Journal daemon for each stream client, as in the worst case the journal daemon needs to buffer the specified number of bytes in memory before it can flush a new log record to disk. Also note that permitting overly large line maximum line lengths affects compatibility with traditional log protocols as log records might not fit anymore into a single **AF_UNIX** or **AF_INET** datagram. Takes a size in bytes. If the value is suffixed with K, M, G or T, the specified size is parsed as Kilobytes, Megabytes, Gigabytes, or Terabytes (with the base 1024), respectively. Defaults to 48K, which is relatively large but still small enough so that log records likely fit into network datagrams along with extra room for metadata. Note that values below 79 are not accepted and will be bumped to 79.

FORWARDING TO TRADITIONAL SYSLOG DAEMONS

Journal events can be transferred to a different logging daemon in two different ways. With the first method, messages are immediately forwarded to a socket (/run/systemd/journal/syslog), where the traditional syslog daemon can read them. This method is controlled by the *ForwardToSyslog=* option. With a second method, a syslog daemon behaves like a normal journal client, and reads messages from the journal files, similarly to **journalctl(1)**. With this, messages do not have to be read immediately, which allows a logging daemon which is only started late in boot to access all messages since the start of the system. In addition, full structured meta-data is available to it. This method of course is available only if the messages are stored in a journal file at all. So it will not work if *Storage=none* is set. It should be noted that usually the *second* method is used by syslog daemons, so the *Storage=* option, and not the *ForwardToSyslog=* option, is relevant for them.

SEE ALSO

systemd(1), **systemd-journald.service(8)**, **journalctl(1)**, **systemd.journal-fields(7)**, **systemd-system.conf(5)**

NOTES

1. Seekable Sequential Key Generators
<https://eprint.iacr.org/2013/397>
2. Users, Groups, UIDs and GIDs on systemd systems
<https://systemd.io/UIDS-GIDS>

NAME

journald.conf, journald.conf.d, journald@.conf – Journal service configuration files

SYNOPSIS

```
/etc/systemd/journald.conf
/etc/systemd/journald.conf.d/*.conf
/run/systemd/journald.conf.d/*.conf
/usr/lib/systemd/journald.conf.d/*.conf
/etc/systemd/journald@NAMESPACE.conf
/etc/systemd/journald@NAMESPACE.conf.d/*.conf
/run/systemd/journald@NAMESPACE.conf.d/*.conf
/usr/lib/systemd/journald@NAMESPACE.conf.d/*.conf
```

DESCRIPTION

These files configure various parameters of the systemd journal service, **systemd-journald.service(8)**. See **systemd.syntax(7)** for a general description of the syntax.

The **systemd-journald** instance managing the default namespace is configured by `/etc/systemd/journald.conf` and associated drop-ins. Instances managing other namespaces read `/etc/systemd/journald@NAMESPACE.conf` and associated drop-ins with the namespace identifier filled in. This allows each namespace to carry a distinct configuration. See **systemd-journald.service(8)** for details about journal namespaces.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in `/etc/systemd/` contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from `/usr/lib/systemd/*.conf.d/`, `/usr/local/lib/systemd/*.conf.d/`, and `/etc/systemd/*.conf.d/`. Those drop-ins have higher precedence and override the main configuration file. Files in the `*.conf.d/` configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under `/usr/`. Files in `/etc/` are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to `/dev/null` in the configuration directory in `/etc/`, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the [Journal] section:

Storage=

Controls where to store journal data. One of "volatile", "persistent", "auto" and "none". If "volatile", journal log data will be stored only in memory, i.e. below the `/run/log/journal` hierarchy (which is created if needed). If "persistent", data will be stored preferably on disk, i.e. below the `/var/log/journal` hierarchy (which is created if needed), with a fallback to `/run/log/journal` (which is created if needed), during early boot and if the disk is not writable. "auto" behaves like "persistent" if the `/var/log/journal` directory exists, and "volatile" otherwise (the existence of the directory controls the storage mode).

"none" turns off all storage, all log data received will be dropped (but forwarding to other targets, such as the console, the kernel log buffer, or a syslog socket will still work). Defaults to "auto" in the default journal namespace, and "persistent" in all others.

Note that journald will initially use volatile storage, until a call to **journalctl --flush** (or sending **SIGUSR1** to journald) will cause it to switch to persistent logging (under the conditions mentioned above). This is done automatically on boot via "systemd-journal-flush.service".

Note that when this option is changed to "volatile", existing persistent data is not removed. In the other direction, **journald(1)** with the **--flush** option may be used to move volatile data to persistent storage.

Compress=

Can take a boolean value. If enabled (the default), data objects that shall be stored in the journal and are larger than the default threshold of 512 bytes are compressed before they are written to the file system. It can also be set to a number of bytes to specify the compression threshold directly. Suffixes like K, M, and G can be used to specify larger units.

Seal=

Takes a boolean value. If enabled (the default), and a sealing key is available (as created by **journald(1)**'s **--setup-keys** command), Forward Secure Sealing (FSS) for all persistent journal files is enabled. FSS is based on [Seekable Sequential Key Generators^{\[1\]}](#) by G. A. Marson and B. Poettering (doi:10.1007/978-3-642-40203-6_7) and may be used to protect journal files from unnoticed alteration.

SplitMode=

Controls whether to split up journal files per user, either "uid" or "none". Split journal files are primarily useful for access control: on UNIX/Linux access control is managed per file, and the journal daemon will assign users read access to their journal files. If "uid", all regular users (with UID outside the range of system users, dynamic service users, and the nobody user) will each get their own journal files, and system users will log to the system journal. See [Users, Groups, UIDs and GIDs on systemd systems^{\[2\]}](#) for more details about UID ranges. If "none", journal files are not split up by user and all messages are instead stored in the single system journal. In this mode unprivileged users generally do not have access to their own log data. Note that splitting up journal files by user is only available for journals stored persistently. If journals are stored on volatile storage (see *Storage=* above), only a single journal file is used. Defaults to "uid".

RateLimitIntervalSec=, RateLimitBurst=

Configures the rate limiting that is applied to all messages generated on the system. If, in the time interval defined by *RateLimitIntervalSec=*, more messages than specified in *RateLimitBurst=* are logged by a service, all further messages within the interval are dropped until the interval is over. A message about the number of dropped messages is generated. This rate limiting is applied per-service, so that two services which log do not interfere with each other's limits. Defaults to 10000 messages in 30s. The time specification for *RateLimitIntervalSec=* may be specified in the following units: "s", "min", "h", "ms", "us". To turn off any kind of rate limiting, set either value to 0.

Note that the effective rate limit is multiplied by a factor derived from the available free disk space for the journal. Currently, this factor is calculated using the base 2 logarithm.

Table 1. Example *RateLimitBurst=* rate modifications by the available disk space

Available Disk Space	Burst Multiplier
<= 1MB	1
<= 16MB	2
<= 256MB	3
<= 4GB	4
<= 64GB	5
<= 1TB	6

If a service provides rate limits for itself through *LogRateLimitIntervalSec*= and/or *LogRateLimitBurst*= in **systemd.exec(5)**, those values will override the settings specified here.

SystemMaxUse=, *SystemKeepFree*=, *SystemMaxFileSize*=, *SystemMaxFiles*=, *RuntimeMaxUse*=, *RuntimeKeepFree*=, *RuntimeMaxFileSize*=, *RuntimeMaxFiles*=

Enforce size limits on the journal files stored. The options prefixed with "System" apply to the journal files when stored on a persistent file system, more specifically /var/log/journal. The options prefixed with "Runtime" apply to the journal files when stored on a volatile in-memory file system, more specifically /run/log/journal. The former is used only when /var/ is mounted, writable, and the directory /var/log/journal exists. Otherwise, only the latter applies. Note that this means that during early boot and if the administrator disabled persistent logging, only the latter options apply, while the former apply if persistent logging is enabled and the system is fully booted up. **journaldctl** and **systemd-journald** ignore all files with names not ending with ".journal" or ".journal~", so only such files, located in the appropriate directories, are taken into account when calculating current disk usage.

SystemMaxUse= and *RuntimeMaxUse*= control how much disk space the journal may use up at most. *SystemKeepFree*= and *RuntimeKeepFree*= control how much disk space **systemd-journald** shall leave free for other uses. **systemd-journald** will respect both limits and use the smaller of the two values.

The first pair defaults to 10% and the second to 15% of the size of the respective file system, but each value is capped to 4G. If the file system is nearly full and either *SystemKeepFree*= or *RuntimeKeepFree*= are violated when **systemd-journald** is started, the limit will be raised to the percentage that is actually free. This means that if there was enough free space before and journal files were created, and subsequently something else causes the file system to fill up, **journald** will stop using more space, but it will not be removing existing files to reduce the footprint again, either. Also note that only archived files are deleted to reduce the space occupied by journal files. This means that, in effect, there might still be more space used than *SystemMaxUse*= or *RuntimeMaxUse*= limit after a vacuuming operation is complete.

SystemMaxFileSize= and *RuntimeMaxFileSize*= control how large individual journal files may grow at most. This influences the granularity in which disk space is made available through rotation, i.e. deletion of historic data. Defaults to one eighth of the values configured with *SystemMaxUse*= and *RuntimeMaxUse*=, so that usually seven rotated journal files are kept as history.

Specify values in bytes or use K, M, G, T, P, E as units for the specified sizes (equal to 1024, 1024², ... bytes). Note that size limits are enforced synchronously when journal files are extended, and no explicit rotation step triggered by time is needed.

SystemMaxFiles= and *RuntimeMaxFiles*= control how many individual journal files to keep at most. Note that only archived files are deleted to reduce the number of files until this limit is reached; active files will stay around. This means that, in effect, there might still be more journal files around in total than this limit after a vacuuming operation is complete. This setting defaults to 100.

MaxFileSec=

The maximum time to store entries in a single journal file before rotating to the next one. Normally, time-based rotation should not be required as size-based rotation with options such as *SystemMaxFileSize*= should be sufficient to ensure that journal files do not grow without bounds.

However, to ensure that not too much data is lost at once when old journal files are deleted, it might make sense to change this value from the default of one month. Set to 0 to turn off this feature. This setting takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

MaxRetentionSec=

The maximum time to store journal entries. This controls whether journal files containing entries older than the specified time span are deleted. Normally, time-based deletion of old journal files should not be required as size-based deletion with options such as *SystemMaxUse=* should be sufficient to ensure that journal files do not grow without bounds. However, to enforce data retention policies, it might make sense to change this value from the default of 0 (which turns off this feature). This setting also takes time values which may be suffixed with the units "year", "month", "week", "day", "h" or "m" to override the default time unit of seconds.

SyncIntervalSec=

The timeout before synchronizing journal files to disk. After syncing, journal files are placed in the OFFLINE state. Note that syncing is unconditionally done immediately after a log message of priority CRIT, ALERT or EMERG has been logged. This setting hence applies only to messages of the levels ERR, WARNING, NOTICE, INFO, DEBUG. The default timeout is 5 minutes.

ForwardToSyslog=, *ForwardToKMsg=*, *ForwardToConsole=*, *ForwardToWall=*

Control whether log messages received by the journal daemon shall be forwarded to a traditional syslog daemon, to the kernel log buffer (kmsg), to the system console, or sent as wall messages to all logged-in users. These options take boolean arguments. If forwarding to syslog is enabled but nothing reads messages from the socket, forwarding to syslog has no effect. By default, only forwarding to syslog and wall is enabled. These settings may be overridden at boot time with the kernel command line options "systemd.journald.forward_to_syslog", "systemd.journald.forward_to_kmsg", "systemd.journald.forward_to_console", and "systemd.journald.forward_to_wall". If the option name is specified without "=" and the following argument, true is assumed. Otherwise, the argument is parsed as a boolean.

When forwarding to the console, the TTY to log to can be changed with *TTYPath=*, described below.

When forwarding to the kernel log buffer (kmsg), make sure to select a suitably large size for the log buffer, for example by adding "log_buf_len=8M" to the kernel command line. **systemd** will automatically disable kernel's rate-limiting applied to userspace processes (equivalent to setting "printk.devkmsg=on").

MaxLevelStore=, *MaxLevelSyslog=*, *MaxLevelKMsg=*, *MaxLevelConsole=*, *MaxLevelWall=*

Controls the maximum log level of messages that are stored in the journal, forwarded to syslog, kmsg, the console or wall (if that is enabled, see above). As argument, takes one of "emerg", "alert", "crit", "err", "warning", "notice", "info", "debug", or integer values in the range of 0–7 (corresponding to the same levels). Messages equal or below the log level specified are stored/forwarded, messages above are dropped. Defaults to "debug" for *MaxLevelStore=* and *MaxLevelSyslog=*, to ensure that all messages are stored in the journal and forwarded to syslog. Defaults to "notice" for *MaxLevelKMsg=*, "info" for *MaxLevelConsole=*, and "emerg" for *MaxLevelWall=*. These settings may be overridden at boot time with the kernel command line options "systemd.journald.max_level_store=", "systemd.journald.max_level_syslog=", "systemd.journald.max_level_kmsg=", "systemd.journald.max_level_console=", "systemd.journald.max_level_wall=".

ReadKMsg=

Takes a boolean value. If enabled **systemd-journal** processes /dev/kmsg messages generated by the kernel. In the default journal namespace this option is enabled by default, it is disabled in all others.

Audit=

Takes a boolean value. If enabled **systemd-journal** will turn on kernel auditing on start-up. If disabled it will turn it off. If unset it will neither enable nor disable it, leaving the previous state unchanged. Note that this option does not control whether **systemd-journald** collects generated audit

records, it just controls whether it tells the kernel to generate them. This means if another tool turns on auditing even if **systemd-journald** left it off, it will still collect the generated messages. Defaults to off.

TTYPath=

Change the console TTY to use if *ForwardToConsole=yes* is used. Defaults to /dev/console.

LineMax=

The maximum line length to permit when converting stream logs into record logs. When a systemd unit's standard output/error are connected to the journal via a stream socket, the data read is split into individual log records at newline ("\n", ASCII 10) and **NUL** characters. If no such delimiter is read for the specified number of bytes a hard log record boundary is artificially inserted, breaking up overly long lines into multiple log records. Selecting overly large values increases the possible memory usage of the Journal daemon for each stream client, as in the worst case the journal daemon needs to buffer the specified number of bytes in memory before it can flush a new log record to disk. Also note that permitting overly large line maximum line lengths affects compatibility with traditional log protocols as log records might not fit anymore into a single **AF_UNIX** or **AF_INET** datagram. Takes a size in bytes. If the value is suffixed with K, M, G or T, the specified size is parsed as Kilobytes, Megabytes, Gigabytes, or Terabytes (with the base 1024), respectively. Defaults to 48K, which is relatively large but still small enough so that log records likely fit into network datagrams along with extra room for metadata. Note that values below 79 are not accepted and will be bumped to 79.

FORWARDING TO TRADITIONAL SYSLOG DAEMONS

Journal events can be transferred to a different logging daemon in two different ways. With the first method, messages are immediately forwarded to a socket (/run/systemd/journal/syslog), where the traditional syslog daemon can read them. This method is controlled by the *ForwardToSyslog=* option. With a second method, a syslog daemon behaves like a normal journal client, and reads messages from the journal files, similarly to **journalctl(1)**. With this, messages do not have to be read immediately, which allows a logging daemon which is only started late in boot to access all messages since the start of the system. In addition, full structured meta-data is available to it. This method of course is available only if the messages are stored in a journal file at all. So it will not work if *Storage=none* is set. It should be noted that usually the *second* method is used by syslog daemons, so the *Storage=* option, and not the *ForwardToSyslog=* option, is relevant for them.

SEE ALSO

systemd(1), **systemd-journald.service(8)**, **journalctl(1)**, **systemd.journal-fields(7)**, **systemd-system.conf(5)**

NOTES

1. Seekable Sequential Key Generators
<https://eprint.iacr.org/2013/397>
2. Users, Groups, UIDs and GIDs on systemd systems
<https://systemd.io/UIDS-GIDS>

NAME

kill – send a signal to a process

SYNOPSIS

kill [options] <pid> [...]

DESCRIPTION

The default signal for kill is TERM. Use **-l** or **-L** to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: **-9**, **-SIGKILL** or **-KILL**. Negative PID values may be used to choose whole process groups; see the PGID column in ps command output. A PID of **-1** is special; it indicates all processes except the kill process itself and init.

OPTIONS

<pid> [...]

Send signal to every <pid> listed.

-<signal>

-s <signal>

--signal <signal>

Specify the **signal** to be sent. The signal can be specified by using name or number. The behavior of signals is explained in **signal(7)** manual page.

-q, --queue value

Use **sigqueue(3)** rather than **kill(2)** and the value argument is used to specify an integer to be sent with the signal. If the receiving process has installed a handler for this signal using the SA_SIGINFO flag to **sigaction(2)**, then it can obtain this data via the si_value field of the siginfo_t structure.

-l, --list [signal]

List signal names. This option has optional argument, which will convert signal number to signal name, or other way round.

-L, --table

List signal names in a nice table.

NOTES

Your shell (command line interpreter) may have a built-in kill command. You may need to run the command described here as /bin/kill to solve the conflict.

EXAMPLES

kill -9 -1

Kill all processes you can kill.

kill -l 11

Translate number 11 into a signal name.

kill -L List the available signal choices in a nice table.

kill 123 543 2341 3453

Send the default signal, SIGTERM, to all those processes.

SEE ALSO

kill(2), killall(1), nice(1), pkill(1), renice(1), signal(7), sigqueue(3), skill(1)

STANDARDS

This command meets appropriate standards. The **-L** flag is Linux-specific.

AUTHOR

Albert Cahalan <albert@users.sf.net> wrote kill in 1999 to replace a bsutils one that was not standards compliant. The util-linux one might also work correctly.

REPORTING BUGS

Please send bug reports to <procps@freelists.org>

NAME

kill – send signal to a process

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <signal.h>
int kill(pid_t pid, int sig);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
kill():
    _POSIX_C_SOURCE
```

DESCRIPTION

The **kill()** system call can be used to send any signal to any process group or process.

If *pid* is positive, then signal *sig* is sent to the process with the ID specified by *pid*.

If *pid* equals 0, then *sig* is sent to every process in the process group of the calling process.

If *pid* equals -1, then *sig* is sent to every process for which the calling process has permission to send signals, except for process 1 (*init*), but see below.

If *pid* is less than -1, then *sig* is sent to every process in the process group whose ID is *-pid*.

If *sig* is 0, then no signal is sent, but existence and permission checks are still performed; this can be used to check for the existence of a process ID or process group ID that the caller is permitted to signal.

For a process to have permission to send a signal, it must either be privileged (under Linux: have the **CAP_KILL** capability in the user namespace of the target process), or the real or effective user ID of the sending process must equal the real or saved set-user-ID of the target process. In the case of **SIGCONT**, it suffices when the sending and receiving processes belong to the same session. (Historically, the rules were different; see NOTES.)

RETURN VALUE

On success (at least one signal was sent), zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL

An invalid signal was specified.

EPERM

The calling process does not have permission to send the signal to any of the target processes.

ESRCH

The target process or process group does not exist. Note that an existing process might be a zombie, a process that has terminated execution, but has not yet been **wait(2)**ed for.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD.

NOTES

The only signals that can be sent to process ID 1, the *init* process, are those for which *init* has explicitly installed signal handlers. This is done to assure the system is not brought down accidentally.

POSIX.1 requires that *kill(-1,sig)* send *sig* to all processes that the calling process may send signals to, except possibly for some implementation-defined system processes. Linux allows a process to signal itself, but on Linux the call *kill(-1,sig)* does not signal the calling process.

POSIX.1 requires that if a process sends a signal to itself, and the sending thread does not have the signal blocked, and no other thread has it unblocked or is waiting for it in **sigwait(3)**, at least one unblocked signal must be delivered to the sending thread before the **kill()** returns.

Linux notes

Across different kernel versions, Linux has enforced different rules for the permissions required for an unprivileged process to send a signal to another process. In Linux 1.0 to 1.2.2, a signal could be sent if the effective user ID of the sender matched effective user ID of the target, or the real user ID of the sender matched the real user ID of the target. From Linux 1.2.3 until 1.3.77, a signal could be sent if the effective user ID of the sender matched either the real or effective user ID of the target. The current rules, which conform to POSIX.1, were adopted in Linux 1.3.78.

BUGS

In Linux 2.6 up to and including Linux 2.6.7, there was a bug that meant that when sending signals to a process group, **kill()** failed with the error **EPEERM** if the caller did not have permission to send the signal to *any* (rather than *all*) of the members of the process group. Notwithstanding this error return, the signal was still delivered to all of the processes for which the caller had permission to signal.

SEE ALSO

kill(1), **_exit(2)**, **pidfd_send_signal(2)**, **signal(2)**, **tkill(2)**, **exit(3)**, **killpg(3)**, **sigqueue(3)**, **capabilities(7)**, **credentials(7)**, **signal(7)**

NAME

killall – kill processes by name

SYNOPSIS

```
killall [-Z, --context pattern] [-e, --exact] [-g, --process-group] [-i, --interactive] [-n, --ns PID]
[-o, --older-than TIME] [-q, --quiet] [-r, --regexp] [-s, --signal SIGNAL, -SIGNAL] [-u, --user
user] [-v, --verbose] [-w, --wait] [-y, --younger-than TIME] [-I, --ignore-case] [-V, --version]
[--] name ...
killall -l
killall -V, --version
```

DESCRIPTION

killall sends a signal to all processes running any of the specified commands. If no signal name is specified, SIGTERM is sent.

Signals can be specified either by name (e.g. **-HUP** or **-SIGHUP**) or by number (e.g. **-1**) or by option **-s**.

If the command name is not regular expression (option **-r**) and contains a slash (/), processes executing that particular file will be selected for killing, independent of their name.

killall returns a zero return code if at least one process has been killed for each listed command, or no commands were listed and at least one process matched the **-u** and **-Z** search criteria. **killall** returns non-zero otherwise.

A **killall** process never kills itself (but may kill other **killall** processes).

OPTIONS**-e, --exact**

Require an exact match for very long names. If a command name is longer than 15 characters, the full name may be unavailable (i.e. it is swapped out). In this case, **killall** will kill everything that matches within the first 15 characters. With **-e**, such entries are skipped. **killall** prints a message for each skipped entry if **-v** is specified in addition to **-e**.

-I, --ignore-case

Do case insensitive process name match.

-g, --process-group

Kill the process group to which the process belongs. The kill signal is only sent once per group, even if multiple processes belonging to the same process group were found.

-i, --interactive

Interactively ask for confirmation before killing.

-l, --list

List all known signal names.

-n, --ns

Match against the PID namespace of the given PID. The default is to match against all namespaces.

-o, --older-than

Match only processes that are older (started before) the time specified. The time is specified as a float then a unit. The units are s,m,h,d,w,M,y for seconds, minutes, hours, days, weeks, months and years respectively.

-q, --quiet

Do not complain if no processes were killed.

-r, --regexp

Interpret process name pattern as a POSIX extended regular expression, per **regex(3)**.

-s, --signal, -*SIGNAL*

Send this signal instead of SIGTERM.

-u, --user

Kill only processes the specified user owns. Command names are optional.

-v, --verbose

Report if the signal was successfully sent.

-V, --version

Display version information.

-w, --wait

Wait for all killed processes to die. **killall** checks once per second if any of the killed processes still exist and only returns if none are left. Note that **killall** may wait forever if the signal was ignored, had no effect, or if the process stays in zombie state.

-y, --younger-than

Match only processes that are younger (started after) the time specified. The time is specified as a float then a unit. The units are s,m,h,d,w,M,y for seconds, minutes, hours, days, weeks, Months and years respectively.

-Z, --context

Specify security context: kill only processes having security context that match with given extended regular expression pattern. Must precede other arguments on the command line. Command names are optional.

FILES

/proc location of the proc file system

KNOWN BUGS

Killing by file only works for executables that are kept open during execution, i.e. impure executables can't be killed this way.

Be warned that typing **killall name** may not have the desired effect on non-Linux systems, especially when done by a privileged user.

killall -w doesn't detect if a process disappears and is replaced by a new process with the same PID between scans.

If processes change their name, **killall** may not be able to match them correctly.

killall has a limit of names that can be specified on the command line. This figure is the size of an unsigned long integer multiplied by 8. For most 32 bit systems the limit is 32 and similarly for a 64 bit system the limit is usually 64.

SEE ALSO

kill(1), **fuser(1)**, **pgrep(1)**, **pidof(1)**, **pkill(1)**, **ps(1)**, **kill(2)**, **regex(3)**.

NAME

killall5 -- send a signal to all processes.

SYNOPSIS

killall5 **-signalnumber** [**-o omitpid[,omitpid...]**] [**-o omitpid[,omitpid...]**...]

DESCRIPTION

killall5 is the SystemV killall command. It sends a signal to all processes except kernel threads and the processes in its own session, so it won't kill the shell that is running the script it was called from. Its primary (only) use is in the **rc** scripts found in the /etc/init.d directory.

OPTIONS

-o omitpid

Tells *killall5* to omit processes with that process id.

NOTES

killall5 can also be invoked as pidof, which is simply a (symbolic) link to the *killall5* program.

EXIT STATUS

The program return zero if it killed processes. It return 2 if no process were killed, and 1 if it was unable to find any processes (/proc/ is missing).

SEE ALSO

halt(8), **reboot(8)**, **pidof(8)**

AUTHOR

Miquel van Smoorenburg, miquels@cistron.nl

NAME

killpg – send signal to a process group

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <signal.h>
int killpg(int pgrp, int sig);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
killpg():
_XOPEN_SOURCE >= 500
|| /* Since glibc 2.19: */ _DEFAULT_SOURCE
|| /* glibc <= 2.19: */ _BSD_SOURCE
```

DESCRIPTION

killpg() sends the signal *sig* to the process group *pgrp*. Seesignal(7) for a list of signals.

If *pgrp* is 0, **killpg()** sends the signal to the calling process's process group. (POSIX says: if *pgrp* is less than or equal to 1, the behavior is undefined.)

For the permissions required to send a signal to another process, see **kill(2)**.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EINVAL

sig is not a valid signal number.

EPERM

The process does not have permission to send the signal to any of the target processes. For the required permissions, see **kill(2)**.

ESRCH

No process can be found in the process group specified by *pgrp*.

ESRCH

The process group was given as 0 but the sending process does not have a process group.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.4BSD (**killpg()** first appeared in 4BSD).

NOTES

There are various differences between the permission checking in BSD-type systems and System V-type systems. See the POSIX rationale for**kill(3p)**. A difference not mentioned by POSIX concerns the return value **EPERM**: BSD documents that no signal is sent and **EPERM** returned when the permission check failed for at least one target process, while POSIX documents **EPERM** only when the permission check failed for all target processes.

C library/kernel differences

On Linux, **killpg()** is implemented as a library function that makes the call *kill(-pgrp, sig)*.

SEE ALSO

getpgrp(2), **kill(2)**, **signal(2)**, **capabilities(7)**, **credentials(7)**

NAME

less – opposite of more

SYNOPSIS

less –?

less --help

less -V

less --version

less [-[+]aABcCdeEfFgGiIJKLMNOPQRSTUVWXYZ]

[-b space] [-h lines] [-j line] [-k keyfile]

[-{oO} logfile] [-p pattern] [-P prompt] [-t tag]

[-T tagsfile] [-x tab,...] [-y lines] [-z lines]

[-# shift] [+ [+cmd] [--] [filename]...]

(See the OPTIONS section for alternate option syntax with long option names.)

DESCRIPTION

Less is a program similar to *more*(1), but it has many more features. *Less* does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like *vi*(1). *Less* uses termcap (or terminfo on some systems), so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with a caret.)

Commands are based on both *more* and *vi*. Commands may be preceded by a decimal number, called N in the descriptions below. The number is used by some commands, as indicated.

COMMANDS

In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for example ESC-v means the two character sequence "ESCAPE", then "v".

h or H Help: display a summary of these commands. If you forget all the other commands, remember this one.

SPACE or ^V or f or ^F

Scroll forward N lines, default one window (see option –z below). If N is more than the screen size, only the final screenful is displayed. Warning: some systems use ^V as a special literalization character.

z Like SPACE, but if N is specified, it becomes the new window size.

ESC-SPACE

Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

ENTER or RETURN or ^N or e or ^E or j or ^J

Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

d or ^D Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

b or ^B or ESC-v

Scroll backward N lines, default one window (see option –z below). If N is more than the screen size, only the final screenful is displayed.

w Like ESC-v, but if N is specified, it becomes the new window size.

y or ^Y or ^P or k or ^K

Scroll backward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size. Warning: some systems use ^Y as a special job control character.

u or ^U Scroll backward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

J Like **j**, but continues to scroll beyond the end of the file.

K or Y Like **k**, but continues to scroll beyond the beginning of the file.

ESC-) or RIGHTARROW

Scroll horizontally right N characters, default half the screen width (see the **-#** option). If a number N is specified, it becomes the default for future **RIGHTARROW** and **LEFTARROW** commands. While the text is scrolled, it acts as though the **-S** option (chop lines) were in effect.

ESC-(or LEFTARROW

Scroll horizontally left N characters, default half the screen width (see the **-#** option). If a number N is specified, it becomes the default for future **RIGHTARROW** and **LEFTARROW** commands.

ESC-} or ^RIGHTARROW

Scroll horizontally right to show the end of the longest displayed line.

ESC-{ or ^LEFTARROW

Scroll horizontally left back to the first column.

r or ^R or ^L

Repaint the screen.

R Repaint the screen, discarding any buffered input. That is, reload the current file. Useful if the file is changing while it is being viewed.

F Scroll forward, and keep trying to read when the end of file is reached. Normally this command would be used when already at the end of the file. It is a way to monitor the tail of a file which is growing while it is being viewed. (The behavior is similar to the "tail -f" command.) To stop waiting for more data, enter the interrupt character (usually **^C**). On some systems you can also use **^X**.

ESC-F Like **F**, but as soon as a line is found which matches the last search pattern, the terminal bell is rung and forward scrolling stops.

g or < or ESC-<

Go to line N in the file, default 1 (beginning of file). (Warning: this may be slow if N is large.)

G or > or ESC->

Go to line N in the file, default the end of the file. (Warning: this may be slow if N is large, or if N is not specified and standard input, rather than a file, is being read.)

ESC-G Same as **G**, except if no number N is specified and the input is standard input, goes to the last line which is currently buffered.

p or % Go to a position N percent into the file. N should be between 0 and 100, and may contain a decimal point.

P Go to the line containing byte offset N in the file.

{ If a left curly bracket appears in the top line displayed on the screen, the **{** command will go to the matching right curly bracket. The matching right curly bracket is positioned on the bottom line of the screen. If there is more than one left curly bracket on the top line, a number N may be used to specify the N-th bracket on the line.

} If a right curly bracket appears in the bottom line displayed on the screen, the **}** command will go to the matching left curly bracket. The matching left curly bracket is positioned on the top line of the screen. If there is more than one right curly bracket on the top line, a number N may be used to specify the N-th bracket on the line.

(Like **{**, but applies to parentheses rather than curly brackets.

) Like **}**, but applies to parentheses rather than curly brackets.

[Like **{**, but applies to square brackets rather than curly brackets.

] Like }, but applies to square brackets rather than curly brackets.

ESC-^F Followed by two characters, acts like {, but uses the two characters as open and close brackets, respectively. For example, "ESC ^F < >" could be used to go forward to the > which matches the < in the top displayed line.

ESC-^B

Followed by two characters, acts like }, but uses the two characters as open and close brackets, respectively. For example, "ESC ^B < >" could be used to go backward to the < which matches the > in the bottom displayed line.

m Followed by any lowercase or uppercase letter, marks the first displayed line with that letter. If the status column is enabled via the -J option, the status column shows the marked line.

M Acts like m, except the last displayed line is marked rather than the first displayed line.

' (Single quote.) Followed by any lowercase or uppercase letter, returns to the position which was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. Followed by a ^ or \$, jumps to the beginning or end of the file respectively. Marks are preserved when a new file is examined, so the ' command can be used to switch between input files.

^X'X Same as single quote.

ESC-m Followed by any lowercase or uppercase letter, clears the mark identified by that letter.

/pattern Search forward in the file for the N-th line containing the pattern. N defaults to 1. The pattern is a regular expression, as recognized by the regular expression library supplied by your system. The search starts at the first line displayed (but see the -a and -j options, which change this).

Certain characters are special if entered at the beginning of the pattern; they modify the type of search rather than become part of the pattern:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the END of the current file without finding a match, the search continues in the next file in the command line list.

^F or @

Begin the search at the first line of the FIRST file in the command line list, regardless of what is currently displayed on the screen or the settings of the -a or -j options.

^K Highlight any text which matches the pattern on the current screen, but don't move to the first match (KEEP current position).

^R Don't interpret regular expression metacharacters; that is, do a simple textual comparison.

^W WRAP around the current file. That is, if the search reaches the end of the current file without finding a match, the search continues from the first line of the current file up to the line where it started.

?pattern

Search backward in the file for the N-th line containing the pattern. The search starts at the last line displayed (but see the -a and -j options, which change this).

Certain characters are special as in the / command:

^N or ! Search for lines which do NOT match the pattern.

^E or * Search multiple files. That is, if the search reaches the beginning of the current file without finding a match, the search continues in the previous file in the command line list.

^F or @

Begin the search at the last line of the last file in the command line list, regardless of what is currently displayed on the screen or the settings of the -a or -j options.

- ^K** As in forward searches.
- ^R** As in forward searches.
- ^W** WRAP around the current file. That is, if the search reaches the beginning of the current file without finding a match, the search continues from the last line of the current file up to the line where it started.

ESC-/pattern

Same as "/*".

ESC-?pattern

Same as "?*".

- n** Repeat previous search, for N-th line containing the last pattern. If the previous search was modified by ^N, the search is made for the N-th line NOT containing the pattern. If the previous search was modified by ^E, the search continues in the next (or previous) file if not satisfied in the current file. If the previous search was modified by ^R, the search is done without using regular expressions. There is no effect if the previous search was modified by ^F or ^K.

- N** Repeat previous search, but in the reverse direction.

- ESC-n** Repeat previous search, but crossing file boundaries. The effect is as if the previous search were modified by *.

- ESC-N** Repeat previous search, but in the reverse direction and crossing file boundaries.

- ESC-u** Undo search highlighting. Turn off highlighting of strings matching the current search pattern. If highlighting is already off because of a previous ESC-u command, turn highlighting back on. Any search command will also turn highlighting back on. (Highlighting can also be disabled by toggling the -G option; in that case search commands do not turn highlighting back on.)

- ESC-U** Like ESC-u but also clears the saved search pattern. If the status column is enabled via the -J option, this clears all search matches marked in the status column.

&pattern

Display only lines which match the pattern; lines which do not match the pattern are not displayed. If pattern is empty (if you type & immediately followed by ENTER), any filtering is turned off, and all lines are displayed. While filtering is in effect, an ampersand is displayed at the beginning of the prompt, as a reminder that some lines in the file may be hidden. Multiple & commands may be entered, in which case only lines which match all of the patterns will be displayed.

Certain characters are special as in the / command:

^N or ! Display only lines which do NOT match the pattern.

^R Don't interpret regular expression metacharacters; that is, do a simple textual comparison.

:e [filename]

Examine a new file. If the filename is missing, the "current" file (see the :n and :p commands below) from the list of files in the command line is re-examined. A percent sign (%) in the filename is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. However, two consecutive percent signs are simply replaced with a single percent sign. This allows you to enter a filename that contains a percent sign in the name. Similarly, two consecutive pound signs are replaced with a single pound sign. The filename is inserted into the command line list of files so that it can be seen by subsequent :n and :p commands. If the filename consists of several files, they are all inserted into the list of files and the first one is examined. If the filename contains one or more spaces, the entire filename should be enclosed in double quotes (also see the -" option).

^X^V or E

Same as :e. Warning: some systems use ^V as a special literalization character. On such systems, you may not be able to use ^V.

- :n Examine the next file (from the list of files given in the command line). If a number N is specified, the N-th next file is examined.
- :p Examine the previous file in the command line list. If a number N is specified, the N-th previous file is examined.
- :x Examine the first file in the command line list. If a number N is specified, the N-th file in the list is examined.
- :d Remove the current file from the list of files.
- t Go to the next tag, if there were more than one matches for the current tag. See the -t option for more details about tags.
- T Go to the previous tag, if there were more than one matches for the current tag.
- = or ^G or :f Prints some information about the file being viewed, including its name and the line number and byte offset of the bottom line being displayed. If possible, it also prints the length of the file, the number of lines in the file and the percent of the file above the last displayed line.
- Followed by one of the command line option letters (see OPTIONS below), this will change the setting of that option and print a message describing the new setting. If a ^P (CONTROL-P) is entered immediately after the dash, the setting of the option is changed but no message is printed. If the option letter has a numeric value (such as -b or -h), or a string value (such as -P or -t), a new value may be entered after the option letter. If no new value is entered, a message describing the current setting is printed and nothing is changed.
- Like the - command, but takes a long option name (see OPTIONS below) rather than a single option letter. You must press ENTER or RETURN after typing the option name. A ^P immediately after the second dash suppresses printing of a message describing the new setting, as in the - command.
- + Followed by one of the command line option letters this will reset the option to its default setting and print a message describing the new setting. (The "-+X" command does the same thing as "+X" on the command line.) This does not work for string-valued options.
- + Like the --+ command, but takes a long option name rather than a single option letter.
- ! Followed by one of the command line option letters, this will reset the option to the "opposite" of its default setting and print a message describing the new setting. This does not work for numeric or string-valued options.
- ! Like the -! command, but takes a long option name rather than a single option letter.
- _ (Underscore.) Followed by one of the command line option letters, this will print a message describing the current setting of that option. The setting of the option is not changed.
- __ (Double underscore.) Like the _ (underscore) command, but takes a long option name rather than a single option letter. You must press ENTER or RETURN after typing the option name.
- +cmd Causes the specified cmd to be executed each time a new file is examined. For example, +G causes *less* to initially display each file starting at the end rather than the beginning.
- V Prints the version number of *less* being run.
- q or Q or :q or :Q or ZZ Exits *less*.

The following four commands may or may not be valid, depending on your particular installation.

- v Invokes an editor to edit the current file being viewed. The editor is taken from the environment variable VISUAL if defined, or EDITOR if VISUAL is not defined, or defaults to "vi" if neither VISUAL nor EDITOR is defined. See also the discussion of LESSEDIT under the section on PROMPTS below.

! shell-command

Invokes a shell to run the shell-command given. A percent sign (%) in the command is replaced by the name of the current file. A pound sign (#) is replaced by the name of the previously examined file. "!!" repeats the last shell command. "!" with no shell command simply invokes a shell. On Unix systems, the shell is taken from the environment variable SHELL, or defaults to "sh". On MS-DOS and OS/2 systems, the shell is the normal command processor.

| <m> shell-command

<m> represents any mark letter. Pipes a section of the input file to the given shell command. The section of the file to be piped is between the position marked by the letter and the current screen. The entire current screen is included, regardless of whether the marked position is before or after the current screen. <m> may also be ^ or \$ to indicate beginning or end of file respectively. If <m> is . or newline, the current screen is piped.

s filename

Save the input to a file. This only works if the input is a pipe, not an ordinary file.

OPTIONS

Command line options are described below. Most options may be changed while *less* is running, via the "--" command.

Most options may be given in one of two forms: either a dash followed by a single letter, or two dashes followed by a long option name. A long option name may be abbreviated as long as the abbreviation is unambiguous. For example, --quit-at-eof may be abbreviated --quit, but not --qui, since both --quit-at-eof and --quiet begin with --qui. Some long option names are in uppercase, such as --QUIT-AT-EOF, as distinct from --quit-at-eof. Such option names need only have their first letter capitalized; the remainder of the name may be in either case. For example, --Quit-at-eof is equivalent to --QUIT-AT-EOF.

Options are also taken from the environment variable "LESS". For example, to avoid typing "less -options ..." each time *less* is invoked, you might tell *csh*:

`setenv LESS "-options"`

or if you use *sh*:

`LESS="-options"; export LESS`

On MS-DOS, you don't need the quotes, but you should replace any percent signs in the options string by double percent signs.

The environment variable is parsed before the command line, so command line options override the LESS environment variable. If an option appears in the LESS variable, it can be reset to its default value on the command line by beginning the command line option with "-+".

Some options like -k or -D require a string to follow the option letter. The string for that option is considered to end when a dollar sign (\$) is found. For example, you can set two -D options on MS-DOS like this:

`LESS="Dn9.1$Ds4.1"`

If the --use-backslash option appears earlier in the options, then a dollar sign or backslash may be included literally in an option string by preceding it with a backslash. If the --use-backslash option is not in effect, then backslashes are not treated specially, and there is no way to include a dollar sign in the option string.

-? or --help

This option displays a summary of the commands accepted by *less* (the same as the h command). (Depending on how your shell interprets the question mark, it may be necessary to quote the question mark, thus: "-\?".)

-a or --search-skip-screen

By default, forward searches start at the top of the displayed screen and backwards searches start at the bottom of the displayed screen (except for repeated searches invoked by the **n** or **N** commands, which start after or before the "target" line respectively; see the **-j** option for more about the target line). The **-a** option causes forward searches to instead start at the bottom of the screen and backward searches to start at the top of the screen, thus skipping all lines displayed on the screen.

-A or --SEARCH-SKIP-SCREEN

Causes all forward searches (not just non-repeated searches) to start just after the target line, and all backward searches to start just before the target line. Thus, forward searches will skip part of the displayed screen (from the first line up to and including the target line). Similarly backwards searches will skip the displayed screen from the last line up to and including the target line. This was the default behavior in less versions prior to 441.

-bn or --buffers=n

Specifies the amount of buffer space *less* will use for each file, in units of kilobytes (1024 bytes). By default 64 KB of buffer space is used for each file (unless the file is a pipe; see the **-B** option). The **-b** option specifies instead that *n* kilobytes of buffer space should be used for each file. If *n* is **-1**, buffer space is unlimited; that is, the entire file can be read into memory.

-B or --auto-buffers

By default, when data is read from a pipe, buffers are allocated automatically as needed. If a large amount of data is read from the pipe, this can cause a large amount of memory to be allocated. The **-B** option disables this automatic allocation of buffers for pipes, so that only 64 KB (or the amount of space specified by the **-b** option) is used for the pipe. Warning: use of **-B** can result in erroneous display, since only the most recently viewed part of the piped data is kept in memory; any earlier data is lost.

-c or --clear-screen

Causes full screen repaints to be painted from the top line down. By default, full screen repaints are done by scrolling from the bottom of the screen.

-C or --CLEAR-SCREEN

Same as **-c**, for compatibility with older versions of *less*.

-d or --dumb

The **-d** option suppresses the error message normally displayed if the terminal is dumb; that is, lacks some important capability, such as the ability to clear the screen or scroll backward. The **-d** option does not otherwise change the behavior of *less* on a dumb terminal.

-Dxcolor or --color=xcolor

Changes the color of different parts of the displayed text. **x** is a single character which selects the type of text whose color is being set:

- B Binary characters.
- C Control characters.
- E Errors and informational messages.
- M Mark letters in the status column.
- N Line numbers enabled via the **-N** option.
- P Prompts.
- R The rscroll character.
- S Search results.
- W The highlight enabled via the **-w** option.
- d Bold text.

- k Blinking text.
- s Standout text.
- u Underlined text.

The uppercase letters can be used only when the `--use-color` option is enabled. When text color is specified by both an uppercase letter and a lowercase letter, the uppercase letter takes precedence. For example, error messages are normally displayed as standout text. So if both "s" and "E" are given a color, the "E" color applies to error messages, and the "s" color applies to other standout text. The "d" and "u" letters refer to bold and underline text formed by overstriking with backspaces (see the `-u` option), not to text using ANSI escape sequences with the `-R` option.

A lowercase letter may be followed by a `+` to indicate that both the normal format change and the specified color should both be used. For example, `-Dug` displays underlined text as green without underlining; the green color has replaced the usual underline formatting. But `-Du+g` displays underlined text as both green and in underlined format.

color is either a 4-bit color string or an 8-bit color string:

A 4-bit color string is zero, one or two characters, where the first character specifies the foreground color and the second specifies the background color as follows:

- b Blue
- c Cyan
- g Green
- k Black
- m Magenta
- r Red
- w White
- y Yellow

The corresponding upper-case letter denotes a brighter shade of the color. For example, `-DNGk` displays line numbers as bright green text on a black background, and `-DEbR` displays error messages as blue text on a bright red background. If either character is a `"-"` or is omitted, the corresponding color is set to that of normal text.

An 8-bit color string is one or two decimal integers separated by a dot, where the first integer specifies the foreground color and the second specifies the background color. Each integer is a value between 0 and 255 inclusive which selects a "CSI 38;5" color value (see

https://en.wikipedia.org/wiki/ANSI_escape_code#SGR_parameters) If either integer is a `"-"` or is omitted, the corresponding color is set to that of normal text. On MS-DOS versions of *less*, 8-bit color is not supported; instead, decimal values are interpreted as 4-bit CHAR_INFO.Attributes values (see

<https://docs.microsoft.com/en-us/windows/console/char-info-str>).

`-e` or `--quit-at-eof`

Causes *less* to automatically exit the second time it reaches end-of-file. By default, the only way to exit *less* is via the `"q"` command.

`-E` or `--QUIT-AT-EOF`

Causes *less* to automatically exit the first time it reaches end-of-file.

`-f` or `--force`

Forces non-regular files to be opened. (A non-regular file is a directory or a device special file.) Also suppresses the warning message when a binary file is opened. By default, *less* will refuse to open non-regular files. Note that some operating systems will not allow directories to be read, even if `-f` is set.

-F or --quit-if-one-screen

Causes *less* to automatically exit if the entire file can be displayed on the first screen.

-g or --hilite-search

Normally, *less* will highlight ALL strings which match the last search command. The **-g** option changes this behavior to highlight only the particular string which was found by the last search command. This can cause *less* to run somewhat faster than the default.

-G or --HILITE-SEARCH

The **-G** option suppresses all highlighting of strings found by search commands.

-hn or --max-back-scroll=n

Specifies a maximum number of lines to scroll backward. If it is necessary to scroll backward more than *n* lines, the screen is repainted in a forward direction instead. (If the terminal does not have the ability to scroll backward, **-h0** is implied.)

-i or --ignore-case

Causes searches to ignore case; that is, uppercase and lowercase are considered identical. This option is ignored if any uppercase letters appear in the search pattern; in other words, if a pattern contains uppercase letters, then that search does not ignore case.

-I or --IGNORE-CASE

Like **-i**, but searches ignore case even if the pattern contains uppercase letters.

-jn or --jump-target=n

Specifies a line on the screen where the "target" line is to be positioned. The target line is the line specified by any command to search for a pattern, jump to a line number, jump to a file percentage or jump to a tag. The screen line may be specified by a number: the top line on the screen is 1, the next is 2, and so on. The number may be negative to specify a line relative to the bottom of the screen: the bottom line on the screen is -1, the second to the bottom is -2, and so on. Alternately, the screen line may be specified as a fraction of the height of the screen, starting with a decimal point: .5 is in the middle of the screen, .3 is three tenths down from the first line, and so on. If the line is specified as a fraction, the actual line number is recalculated if the terminal window is resized, so that the target line remains at the specified fraction of the screen height. If any form of the **-j** option is used, repeated forward searches (invoked with "n" or "N") begin at the line immediately after the target line, and repeated backward searches begin at the target line, unless changed by **-a** or **-A**. For example, if "**-j4**" is used, the target line is the fourth line on the screen, so forward searches begin at the fifth line on the screen. However nonrepeated searches (invoked with "/" or "?") always begin at the start or end of the current screen respectively.

-J or --status-column

Displays a status column at the left edge of the screen. The status column shows the lines that matched the current search, and any lines that are marked (via the **m** or **M** command).

-kfilename or --lesskey-file=filename

Causes *less* to open and interpret the named file as a *lesskey*(1) binary file. Multiple **-k** options may be specified. If the LESSKEY or LESSKEY_SYSTEM environment variable is set, or if a lesskey file is found in a standard place (see KEY BINDINGS), it is also used as a *lesskey* file.

--lesskey-src=filename

Causes *less* to open and interpret the named file as a *lesskey*(1) source file. If the LESSKEYIN or LESSKEYIN_SYSTEM environment variable is set, or if a lesskey source file is found in a standard place (see KEY BINDINGS), it is also used as a *lesskey source* file. Prior to version 582, the *lesskey* program needed to be run to convert a *lesskey source* file to a *lesskey binary* file for *less* to use. Newer versions of *less* read the *lesskey source* file directly and ignore the binary file if the source file exists.

-K or --quit-on-intr

Causes *less* to exit immediately (with status 2) when an interrupt character (usually ^C) is typed. Normally, an interrupt character causes *less* to stop whatever it is doing and return to its command

prompt. Note that use of this option makes it impossible to return to the command prompt from the "F" command.

-L or --no-lessopen

Ignore the LESSOPEN environment variable (see the INPUT PREPROCESSOR section below). This option can be set from within *less*, but it will apply only to files opened subsequently, not to the file which is currently open.

-m or --long-prompt

Causes *less* to prompt verbosely (like *more*), with the percent into the file. By default, *less* prompts with a colon.

-M or --LONG-PROMPT

Causes *less* to prompt even more verbosely than *more*.

-n or --line-numbers

Suppresses line numbers. The default (to use line numbers) may cause *less* to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with the -n option will avoid this problem. Using line numbers means: the line number will be displayed in the verbose prompt and in the = command, and the v command will pass the current line number to the editor (see also the discussion of LESSEDIT in PROMPTS below).

-N or --LINE-NUMBERS

Causes a line number to be displayed at the beginning of each line in the display.

-filename or --log-file=filename

Causes *less* to copy its input to the named file as it is being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, *less* will ask for confirmation before overwriting it.

-Ofilename or --LOG-FILE=filename

The -O option is like -o, but it will overwrite an existing file without asking for confirmation.

If no log file has been specified, the -o and -O options can be used from within *less* to specify a log file. Without a file name, they will simply report the name of the log file. The "s" command is equivalent to specifying -o from within *less*.

-ppattern or --pattern=pattern

The -p option on the command line is equivalent to specifying +/pattern; that is, it tells *less* to start at the first occurrence of pattern in the file.

-Pprompt or --prompt=prompt

Provides a way to tailor the three prompt styles to your own preference. This option would normally be put in the LESS environment variable, rather than being typed in with each *less* command. Such an option must either be the last option in the LESS variable, or be terminated by a dollar sign.

-Ps followed by a string changes the default (short) prompt to that string.

-Pm changes the medium (-m) prompt.

-PM changes the long (-M) prompt.

-Ph changes the prompt for the help screen.

-P= changes the message printed by the = command.

-Pw changes the message printed while waiting for data (in the F command).

All prompt strings consist of a sequence of letters and special escape sequences. See the section on PROMPTS for more details.

-q or --quiet or --silent

Causes moderately "quiet" operation: the terminal bell is not rung if an attempt is made to scroll past the end of the file or before the beginning of the file. If the terminal has a "visual bell", it is used instead. The bell will be rung on certain other errors, such as typing an invalid character.

The default is to ring the terminal bell in all such cases.

-Q or --QUIET or --SILENT

Causes totally "quiet" operation: the terminal bell is never rung. If the terminal has a "visual bell", it is used in all cases where the terminal bell would have been rung.

-r or --raw-control-chars

Causes "raw" control characters to be displayed. The default is to display control characters using the caret notation; for example, a control-A (octal 001) is displayed as "^A". Warning: when the -r option is used, *less* cannot keep track of the actual appearance of the screen (since this depends on how the screen responds to each type of control character). Thus, various display problems may result, such as long lines being split in the wrong place.

USE OF THE -r OPTION IS NOT RECOMMENDED.

-R or --RAW-CONTROL-CHARS

Like -r, but only ANSI "color" escape sequences and OSC 8 hyperlink sequences are output in "raw" form. Unlike -r, the screen appearance is maintained correctly, provided that there are no escape sequences in the file other than these types of escape sequences. Color escape sequences are only supported when the color is changed within one line, not across lines. In other words, the beginning of each line is assumed to be normal (non-colored), regardless of any escape sequences in previous lines. For the purpose of keeping track of screen appearance, these escape sequences are assumed to not move the cursor.

OSC 8 hyperlinks are sequences of the form:

ESC] 8 ; ... \7

The terminating sequence may be either a BEL character (\7) or the two-character sequence "ESC \".

ANSI color escape sequences are sequences of the form:

ESC [... m

where the "..." is zero or more color specification characters. You can make *less* think that characters other than "m" can end ANSI color escape sequences by setting the environment variable LESSANSIENDCHARS to the list of characters which can end a color escape sequence. And you can make *less* think that characters other than the standard ones may appear between the ESC and the m by setting the environment variable LESSANSIMIDCHARS to the list of characters which can appear.

-s or --squeeze-blank-lines

Causes consecutive blank lines to be squeezed into a single blank line. This is useful when viewing *nroff* output.

-S or --chop-long-lines

Causes lines longer than the screen width to be chopped (truncated) rather than wrapped. That is, the portion of a long line that does not fit in the screen width is not displayed until you press RIGHT-ARROW. The default is to wrap long lines; that is, display the remainder on the next line.

-ttag or --tag=tag

The -t option, followed immediately by a TAG, will edit the file containing that tag. For this to work, tag information must be available; for example, there may be a file in the current directory called "tags", which was previously built by *ctags(1)* or an equivalent command. If the environment variable LESSGLOBALTAGS is set, it is taken to be the name of a command compatible with *global(1)*, and that command is executed to find the tag. (See <http://www.gnu.org/software/global/global.html>). The -t option may also be specified from within

less (using the **-** command) as a way of examining a new file. The command ":t" is equivalent to specifying **-t** from within *less*.

-T*tagsfile* or **--tag-file=tagsfile**

Specifies a tags file to be used instead of "tags".

-u or **--underline-special**

Causes backspaces and carriage returns to be treated as printable characters; that is, they are sent to the terminal when they appear in the input.

-U or **--UNDERLINE-SPECIAL**

Causes backspaces, tabs, carriage returns and "formatting characters" (as defined by Unicode) to be treated as control characters; that is, they are handled as specified by the **-r** option.

By default, if neither **-u** nor **-U** is given, backspaces which appear adjacent to an underscore character are treated specially: the underlined text is displayed using the terminal's hardware underlining capability. Also, backspaces which appear between two identical characters are treated specially: the overstruck text is printed using the terminal's hardware boldface capability. Other backspaces are deleted, along with the preceding character. Carriage returns immediately followed by a newline are deleted. Other carriage returns are handled as specified by the **-r** option. Unicode formatting characters, such as the Byte Order Mark, are sent to the terminal. Text which is overstruck or underlined can be searched for if neither **-u** nor **-U** is in effect.

-V or **--version**

Displays the version number of *less*.

-w or **--hilite-unread**

Temporarily highlights the first "new" line after a forward movement of a full page. The first "new" line is the line immediately following the line previously at the bottom of the screen. Also highlights the target line after a **g** or **p** command. The highlight is removed at the next command which causes movement. The entire line is highlighted, unless the **-J** option is in effect, in which case only the status column is highlighted.

-W or **--HILITE-UNREAD**

Like **-w**, but temporarily highlights the first new line after any forward movement command larger than one line.

-xn,... or **--tabs=n,...**

Sets tab stops. If only one *n* is specified, tab stops are set at multiples of *n*. If multiple values separated by commas are specified, tab stops are set at those positions, and then continue with the same spacing as the last two. For example, **-x9,17** will set tabs at positions 9, 17, 25, 33, etc. The default for *n* is 8.

-X or **--no-init**

Disables sending the termcap initialization and deinitialization strings to the terminal. This is sometimes desirable if the deinitialization string does something unnecessary, like clearing the screen.

-yn or **--max-forw-scroll=n**

Specifies a maximum number of lines to scroll forward. If it is necessary to scroll forward more than *n* lines, the screen is repainted instead. The **-c** or **-C** option may be used to repaint from the top of the screen if desired. By default, any forward movement causes scrolling.

-zn or **--window=n** or **-n**

Changes the default scrolling window size to *n* lines. The default is one screenful. The **z** and **w** commands can also be used to change the window size. The "z" may be omitted for compatibility with some versions of *more*. If the number *n* is negative, it indicates *n* lines less than the current screen size. For example, if the screen is 24 lines, **-z-4** sets the scrolling window to 20 lines. If the screen is resized to 40 lines, the scrolling window automatically changes to 36 lines.

–"cc or --quotes=cc

Changes the filename quoting character. This may be necessary if you are trying to name a file which contains both spaces and quote characters. Followed by a single character, this changes the quote character to that character. Filenames containing a space should then be surrounded by that character rather than by double quotes. Followed by two characters, changes the open quote to the first character, and the close quote to the second character. Filenames containing a space should then be preceded by the open quote character and followed by the close quote character. Note that even after the quote characters are changed, this option remains –" (a dash followed by a double quote).

–~ or --tilde

Normally lines after end of file are displayed as a single tilde (~). This option causes lines after end of file to be displayed as blank lines.

–# or --shift

Specifies the default number of positions to scroll horizontally in the RIGHTARROW and LEFT-ARROW commands. If the number specified is zero, it sets the default number of positions to one half of the screen width. Alternately, the number may be specified as a fraction of the width of the screen, starting with a decimal point: .5 is half of the screen width, .3 is three tenths of the screen width, and so on. If the number is specified as a fraction, the actual number of scroll positions is recalculated if the terminal window is resized, so that the actual scroll remains at the specified fraction of the screen width.

--file-size

If --file-size is specified, *less* will determine the size of the file immediately after opening the file. Normally this is not done, because it can be slow if the input file is large.

--follow-name

Normally, if the input file is renamed while an F command is executing, *less* will continue to display the contents of the original file despite its name change. If --follow-name is specified, during an F command *less* will periodically attempt to reopen the file by name. If the reopen succeeds and the file is a different file from the original (which means that a new file has been created with the same name as the original (now renamed) file), *less* will display the contents of that new file.

--incsearch

Subsequent search commands will be "incremental"; that is, *less* will advance to the next line containing the search pattern as each character of the pattern is typed in.

--line-num-width

Sets the minimum width of the line number field when the –N option is in effect. The default is 7 characters.

--mouse

Enables mouse input: scrolling the mouse wheel down moves forward in the file, scrolling the mouse wheel up moves backwards in the file, and clicking the mouse sets the "#" mark to the line where the mouse is clicked. The number of lines to scroll when the wheel is moved can be set by the --wheel-lines option. Mouse input works only on terminals which support X11 mouse reporting, and on the Windows version of *less*.

--MOUSE

Like --mouse, except the direction scrolled on mouse wheel movement is reversed.

--no-keypad

Disables sending the keypad initialization and deinitialization strings to the terminal. This is sometimes useful if the keypad strings make the numeric keypad behave in an undesirable manner.

--no-histdups

This option changes the behavior so that if a search string or file name is typed in, and the same string is already in the history list, the existing copy is removed from the history list before the

new one is added. Thus, a given string will appear only once in the history list. Normally, a string may appear multiple times.

--rscroll

This option changes the character used to mark truncated lines. It may begin with a two-character attribute indicator like LESSBINFMT does. If there is no attribute indicator, standout is used. If set to "-", truncated lines are not marked.

--save-marks

Save marks in the history file, so marks are retained across different invocations of *less*.

--status-col-width

Sets the width of the status column when the -J option is in effect. The default is 2 characters.

--use-backslash

This option changes the interpretations of options which follow this one. After the --use-backslash option, any backslash in an option string is removed and the following character is taken literally. This allows a dollar sign to be included in option strings.

--use-color

Enables the colored text in various places. The -D option can be used to change the colors. Colored text works only if the terminal supports ANSI color escape sequences (as defined in ECMA-48 SGR; see

<https://www.ecma-international.org/publications-and-standards/standards/ecma-48>).

--wheel-lines=n

Set the number of lines to scroll when the mouse wheel is scrolled and the --mouse or --MOUSE option is in effect. The default is 1 line.

-- A command line argument of "--" marks the end of option arguments. Any arguments following this are interpreted as filenames. This can be useful when viewing a file whose name begins with a "--" or "+".

+ If a command line option begins with +, the remainder of that option is taken to be an initial command to *less*. For example, +G tells *less* to start at the end of the file rather than the beginning, and +/xyz tells it to start at the first occurrence of "xyz" in the file. As a special case, +<number> acts like +<number>g; that is, it starts the display at the specified line number (however, see the caveat under the "g" command above). If the option starts with ++, the initial command applies to every file being viewed, not just the first one. The + command described previously may also be used to set (or change) an initial command for every file.

LINE EDITING

When entering a command line at the bottom of the screen (for example, a filename for the :e command, or the pattern for a search command), certain keys can be used to manipulate the command line. Most commands have an alternate form in [brackets] which can be used if a key does not exist on a particular keyboard. (Note that the forms beginning with ESC do not work in some MS-DOS and Windows systems because ESC is the line erase character.) Any of these special keys may be entered literally by preceding it with the "literal" character, either ^V or ^A. A backslash itself may also be entered literally by entering two backslashes.

LEFTARROW [ESC-h]

Move the cursor one space to the left.

RIGHTARROW [ESC-1]

Move the cursor one space to the right.

^LEFTARROW [ESC-b or ESC-LEFTARROW]

(That is, CONTROL and LEFTARROW simultaneously.) Move the cursor one word to the left.

^RIGHTARROW [ESC-w or ESC-RIGHTARROW]

(That is, CONTROL and RIGHTARROW simultaneously.) Move the cursor one word to the right.

- HOME [ESC-0]**
Move the cursor to the beginning of the line.
- END [ESC-\$]**
Move the cursor to the end of the line.
- BACKSPACE**
Delete the character to the left of the cursor, or cancel the command if the command line is empty.
- DELETE or [ESC-x]**
Delete the character under the cursor.
- ^BACKSPACE [ESC-BACKSPACE]**
(That is, CONTROL and BACKSPACE simultaneously.) Delete the word to the left of the cursor.
- ^DELETE [ESC-X or ESC-DELETE]**
(That is, CONTROL and DELETE simultaneously.) Delete the word under the cursor.
- UPARROW [ESC-k]**
Retrieve the previous command line. If you first enter some text and then press UPARROW, it will retrieve the previous command which begins with that text.
- DOWNARROW [ESC-j]**
Retrieve the next command line. If you first enter some text and then press DOWNARROW, it will retrieve the next command which begins with that text.
- TAB** Complete the partial filename to the left of the cursor. If it matches more than one filename, the first match is entered into the command line. Repeated TABs will cycle thru the other matching filenames. If the completed filename is a directory, a "/" is appended to the filename. (On MS-DOS systems, a "\" is appended.) The environment variable LESSSEPARATOR can be used to specify a different character to append to a directory name.
- BACKTAB [ESC-TAB]**
Like, TAB, but cycles in the reverse direction thru the matching filenames.
- ^L** Complete the partial filename to the left of the cursor. If it matches more than one filename, all matches are entered into the command line (if they fit).
- ^U (Unix and OS/2) or ESC (MS-DOS)**
Delete the entire command line, or cancel the command if the command line is empty. If you have changed your line-kill character in Unix to something other than ^U, that character is used instead of ^U.
- ^G** Delete the entire command line and return to the main prompt.

KEY BINDINGS

You may define your own *less* commands by creating a lesskey source file. This file specifies a set of command keys and an action associated with each key. You may also change the line-editing keys (see LINE EDITING), and to set environment variables. If the environment variable LESSKEYIN is set, *less* uses that as the name of the lesskey source file. Otherwise, *less* looks in a standard place for the lesskey source file: On Unix systems, *less* looks for a lesskey file called "\$XDG_CONFIG_HOME/lesskey" or "\$HOME/.lesskey". On MS-DOS and Windows systems, *less* looks for a lesskey file called "\$HOME/_lesskey", and if it is not found there, then looks for a lesskey file called "_lesskey" in any directory specified in the PATH environment variable. On OS/2 systems, *less* looks for a lesskey file called "\$HOME/lesskey.ini", and if it is not found, then looks for a lesskey file called "lesskey.ini" in any directory specified in the INIT environment variable, and if it not found there, then looks for a lesskey file called "lesskey.ini" in any directory specified in the PATH environment variable. See the lesskey manual page for more details.

A system-wide lesskey source file may also be set up to provide key bindings. If a key is defined in both a local lesskey file and in the system-wide file, key bindings in the local file take precedence over those in the system-wide file. If the environment variable LESSKEYIN_SYSTEM is set, *less* uses that as the name of the system-wide lesskey file. Otherwise, *less* looks in a standard place for the system-wide lesskey file: On

Unix systems, the system-wide lesskey file is /usr/local/etc/syslesskey. (However, if *less* was built with a different sysconf directory than /usr/local/etc, that directory is where the sysless file is found.) On MS-DOS and Windows systems, the system-wide lesskey file is c:\syslesskey. On OS/2 systems, the system-wide lesskey file is c:\syslesskey.ini.

Previous versions of *less* (before v582) used lesskey files with a binary format, produced by the *lesskey* program. It is no longer necessary to use the *lesskey* program.

INPUT PREPROCESSOR

You may define an "input preprocessor" for *less*. Before *less* opens a file, it first gives your input preprocessor a chance to modify the way the contents of the file are displayed. An input preprocessor is simply an executable program (or shell script), which writes the contents of the file to a different file, called the replacement file. The contents of the replacement file are then displayed in place of the contents of the original file. However, it will appear to the user as if the original file is opened; that is, *less* will display the original filename as the name of the current file.

An input preprocessor receives one command line argument, the original filename, as entered by the user. It should create the replacement file, and when finished, print the name of the replacement file to its standard output. If the input preprocessor does not output a replacement filename, *less* uses the original file, as normal. The input preprocessor is not called when viewing standard input. To set up an input preprocessor, set the LESSOPEN environment variable to a command line which will invoke your input preprocessor. This command line should include one occurrence of the string "%s", which will be replaced by the filename when the input preprocessor command is invoked.

When *less* closes a file opened in such a way, it will call another program, called the input postprocessor, which may perform any desired clean-up action (such as deleting the replacement file created by LESSOPEN). This program receives two command line arguments, the original filename as entered by the user, and the name of the replacement file. To set up an input postprocessor, set the LESSCLOSE environment variable to a command line which will invoke your input postprocessor. It may include two occurrences of the string "%s"; the first is replaced with the original name of the file and the second with the name of the replacement file, which was output by LESSOPEN.

For example, on many Unix systems, these two scripts will allow you to keep files in compressed format, but still let *less* view them directly:

```
lessopen.sh:
#!/bin/sh
case "$1" in
*.Z)   TEMPFILE=$(mktemp)
       uncompress -c $1 >$TEMPFILE 2>/dev/null
       if [ -s $TEMPFILE ]; then
           echo $TEMPFILE
       else
           rm -f $TEMPFILE
       fi
       ;;
esac

lessclose.sh:
#!/bin/sh
rm $2
```

To use these scripts, put them both where they can be executed and set LESSOPEN="lessopen.sh %s", and LESSCLOSE="lessclose.sh %s %s". More complex LESSOPEN and LESSCLOSE scripts may be written to accept other types of compressed files, and so on.

It is also possible to set up an input preprocessor to pipe the file data directly to *less*, rather than putting the data into a replacement file. This avoids the need to decompress the entire file before starting to view it. An input preprocessor that works this way is called an input pipe. An input pipe, instead of writing the name of a replacement file on its standard output, writes the entire contents of the replacement file on its

standard output. If the input pipe does not write any characters on its standard output, then there is no replacement file and *less* uses the original file, as normal. To use an input pipe, make the first character in the LESSOPEN environment variable a vertical bar (|) to signify that the input preprocessor is an input pipe. As with non-pipe input preprocessors, the command string must contain one occurrence of %s, which is replaced with the filename of the input file.

For example, on many Unix systems, this script will work like the previous example scripts:

`lesspipe.sh`:

```
#!/bin/sh
case "$1" in
*.Z)    uncompress -c $1 2>/dev/null
;;
*)      exit 1
;;
esac
exit $?
```

To use this script, put it where it can be executed and set LESSOPEN="|lesspipe.sh %s".

Note that a preprocessor cannot output an empty file, since that is interpreted as meaning there is no replacement, and the original file is used. To avoid this, if LESSOPEN starts with two vertical bars, the exit status of the script becomes meaningful. If the exit status is zero, the output is considered to be replacement text, even if it is empty. If the exit status is nonzero, any output is ignored and the original file is used. For compatibility with previous versions of *less*, if LESSOPEN starts with only one vertical bar, the exit status of the preprocessor is ignored.

When an input pipe is used, a LESSCLOSE postprocessor can be used, but it is usually not necessary since there is no replacement file to clean up. In this case, the replacement file name passed to the LESSCLOSE postprocessor is "-".

For compatibility with previous versions of *less*, the input preprocessor or pipe is not used if *less* is viewing standard input. However, if the first character of LESSOPEN is a dash (-), the input preprocessor is used on standard input as well as other files. In this case, the dash is not considered to be part of the preprocessor command. If standard input is being viewed, the input preprocessor is passed a file name consisting of a single dash. Similarly, if the first two characters of LESSOPEN are vertical bar and dash (|-) or two vertical bars and a dash (||-), the input pipe is used on standard input as well as other files. Again, in this case the dash is not considered to be part of the input pipe command.

NATIONAL CHARACTER SETS

There are three types of characters in the input file:

normal characters

can be displayed directly to the screen.

control characters

should not be displayed directly, but are expected to be found in ordinary text files (such as backspace and tab).

binary characters

should not be displayed directly and are not expected to be found in text files.

A "character set" is simply a description of which characters are to be considered normal, control, and binary. The LESSCHARSET environment variable may be used to select a character set. Possible values for LESSCHARSET are:

ascii BS, TAB, NL, CR, and formfeed are control characters, all chars with values between 32 and 126 are normal, and all others are binary.

iso8859

Selects an ISO 8859 character set. This is the same as ASCII, except characters between 160 and 255 are treated as normal characters.

latin1	Same as iso8859.
latin9	Same as iso8859.
dos	Selects a character set appropriate for MS-DOS.
ebcdic	Selects an EBCDIC character set.
IBM-1047	Selects an EBCDIC character set used by OS/390 Unix Services. This is the EBCDIC analogue of latin1. You get similar results by setting either LESSCHARSET=IBM-1047 or LC_CTYPE=en_US in your environment.
koi8-r	Selects a Russian character set.
next	Selects a character set appropriate for NeXT computers.
utf-8	Selects the UTF-8 encoding of the ISO 10646 character set. UTF-8 is special in that it supports multi-byte characters in the input file. It is the only character set that supports multi-byte characters.
windows	Selects a character set appropriate for Microsoft Windows (cp 1251).

In rare cases, it may be desired to tailor *less* to use a character set other than the ones definable by LESS-CHARSET. In this case, the environment variable LESSCHARDEF can be used to define a character set. It should be set to a string where each character in the string represents one character in the character set. The character "." is used for a normal character, "c" for control, and "b" for binary. A decimal number may be used for repetition. For example, "bccc4b." would mean character 0 is binary, 1, 2 and 3 are control, 4, 5, 6 and 7 are binary, and 8 is normal. All characters after the last are taken to be the same as the last, so characters 9 through 255 would be normal. (This is an example, and does not necessarily represent any real character set.)

This table shows the value of LESSCHARDEF which is equivalent to each of the possible values for LESS-CHARSET:

ascii	8bcccbbc18b95.b
dos	8bcccbbc12bc5b95.b.
ebcdic	5bc6bcc7bcc41b.9b7.9b5.b..8b6.10b6.b9.7b 9.8b8.17b3.3b9.7b9.8b8.6b10.b.b.b.
IBM-1047	4cbcbbc3b9cbccbccbb4c6bcc5b3cbcc4bc4bccbc 191.b
iso8859	8bcccbbc18b95.33b.
koi8-r	8bcccbbc18b95.b128.
latin1	8bcccbbc18b95.33b.
next	8bcccbbc18b95.bb125.bb

If neither LESSCHARSET nor LESSCHARDEF is set, but any of the strings "UTF-8", "UTF8", "utf-8" or "utf8" is found in the LC_ALL, LC_CTYPE or LANG environment variables, then the default character set is utf-8.

If that string is not found, but your system supports the *setlocale* interface, *less* will use *setlocale* to determine the character set. *setlocale* is controlled by setting the LANG or LC_CTYPE environment variables.

Finally, if the *setlocale* interface is also not available, the default character set is latin1.

Control and binary characters are displayed in standout (reverse video). Each such character is displayed in caret notation if possible (e.g. ^A for control-A). Caret notation is used only if inverting the 0100 bit results in a normal printable character. Otherwise, the character is displayed as a hex number in angle brackets. This format can be changed by setting the LESSBINFMT environment variable. LESSBINFMT may begin with a "*" and one character to select the display attribute: "*k" is blinking, "*d" is bold, "*u" is underlined, "*s" is standout, and "*n" is normal. If LESSBINFMT does not begin with a "*", normal attribute is assumed. The remainder of LESSBINFMT is a string which may include one printf-style escape sequence (a

% followed by x, X, o, d, etc.). For example, if LESSBINFMT is "*u[%x]", binary characters are displayed in underlined hexadecimal surrounded by brackets. The default if no LESSBINFMT is specified is "*s<%02X>". Warning: the result of expanding the character via LESSBINFMT must be less than 31 characters.

When the character set is utf-8, the LESSUTFBINFMT environment variable acts similarly to LESSBINFMT but it applies to Unicode code points that were successfully decoded but are unsuitable for display (e.g., unassigned code points). Its default value is "<U+%04IX>". Note that LESSUTFBINFMT and LESSBINFMT share their display attribute setting ("*x") so specifying one will affect both; LESSUTFBINFMT is read after LESSBINFMT so its setting, if any, will have priority. Problematic octets in a UTF-8 file (octets of a truncated sequence, octets of a complete but non-shortest form sequence, invalid octets, and stray trailing octets) are displayed individually using LESSBINFMT so as to facilitate diagnostic of how the UTF-8 file is ill-formed.

PROMPTS

The -P option allows you to tailor the prompt to your preference. The string given to the -P option replaces the specified prompt string. Certain characters in the string are interpreted specially. The prompt mechanism is rather complicated to provide flexibility, but the ordinary user need not understand the details of constructing personalized prompt strings.

A percent sign followed by a single character is expanded according to what the following character is:

- %bX Replaced by the byte offset into the current input file. The b is followed by a single character (shown as X above) which specifies the line whose byte offset is to be used. If the character is a "t", the byte offset of the top line in the display is used, an "m" means use the middle line, a "b" means use the bottom line, a "B" means use the line just after the bottom line, and a "j" means use the "target" line, as specified by the -j option.
- %B Replaced by the size of the current input file.
- %c Replaced by the column number of the text appearing in the first column of the screen.
- %dX Replaced by the page number of a line in the input file. The line to be used is determined by the X, as with the %b option.
- %D Replaced by the number of pages in the input file, or equivalently, the page number of the last line in the input file.
- %E Replaced by the name of the editor (from the VISUAL environment variable, or the EDITOR environment variable if VISUAL is not defined). See the discussion of the LESSEDIT feature below.
- %f Replaced by the name of the current input file.
- %F Replaced by the last component of the name of the current input file.
- %g Replaced by the shell-escaped name of the current input file. This is useful when the expanded string will be used in a shell command, such as in LESSEDIT.
- %i Replaced by the index of the current file in the list of input files.
- %lX Replaced by the line number of a line in the input file. The line to be used is determined by the X, as with the %b option.
- %L Replaced by the line number of the last line in the input file.
- %m Replaced by the total number of input files.
- %pX Replaced by the percent into the current input file, based on byte offsets. The line used is determined by the X as with the %b option.
- %PX Replaced by the percent into the current input file, based on line numbers. The line used is determined by the X as with the %b option.
- %s Same as %B.

- %t Causes any trailing spaces to be removed. Usually used at the end of the string, but may appear anywhere.
- %T Normally expands to the word "file". However if viewing files via a tags list using the -t option, it expands to the word "tag".
- %x Replaced by the name of the next input file in the list.

If any item is unknown (for example, the file size if input is a pipe), a question mark is printed instead.

The format of the prompt string can be changed depending on certain conditions. A question mark followed by a single character acts like an "IF": depending on the following character, a condition is evaluated. If the condition is true, any characters following the question mark and condition character, up to a period, are included in the prompt. If the condition is false, such characters are not included. A colon appearing between the question mark and the period can be used to establish an "ELSE": any characters between the colon and the period are included in the string if and only if the IF condition is false. Condition characters (which follow a question mark) may be:

- ?a True if any characters have been included in the prompt so far.
- ?bX True if the byte offset of the specified line is known.
- ?B True if the size of current input file is known.
- ?c True if the text is horizontally shifted (%c is not zero).
- ?dX True if the page number of the specified line is known.
- ?e True if at end-of-file.
- ?f True if there is an input filename (that is, if input is not a pipe).
- ?lX True if the line number of the specified line is known.
- ?L True if the line number of the last line in the file is known.
- ?m True if there is more than one input file.
- ?n True if this is the first prompt in a new input file.
- ?pX True if the percent into the current input file, based on byte offsets, of the specified line is known.
- ?PX True if the percent into the current input file, based on line numbers, of the specified line is known.
- ?s Same as "?B".
- ?x True if there is a next input file (that is, if the current input file is not the last one).

Any characters other than the special ones (question mark, colon, period, percent, and backslash) become literally part of the prompt. Any of the special characters may be included in the prompt literally by preceding it with a backslash.

Some examples:

?f%f:Standard input.

This prompt prints the filename, if known; otherwise the string "Standard input".

?f%f .?lLine %lt:?pt%pt\%:?btByte %bt:-...

This prompt would print the filename, if known. The filename is followed by the line number, if known, otherwise the percent if known, otherwise the byte offset if known. Otherwise, a dash is printed. Notice how each question mark has a matching period, and how the % after the %pt is included literally by escaping it with a backslash.

?n?f%f .?m(%T %i of %m) ..?e(END) ?x- Next\?: %x..%t";

This prints the filename if this is the first prompt in a file, followed by the "file N of N" message if there is more than one input file. Then, if we are at end-of-file, the string "(END)" is printed followed by the name of the next file, if there is one. Finally, any trailing spaces are truncated. This is the default prompt. For reference, here are the defaults for the other two prompts (`-m` and `-M` respectively). Each is broken into two lines here for readability only.

```
?n?f%f .?m(%T %i of %m) ..?e(END) ?x- Next\!: %x.:  
?pB%pB\%:byte %bB?s/%s...%t  
  
?f%f .?n?m(%T %i of %m) ..?lrlines %lt-%lb?L/%L. :  
byte %bB?s/%s..?e(END) ?x- Next\!: %x.:?pB%pB\%..%t
```

And here is the default message produced by the `=` command:

```
?f%f .?m(%T %i of %m) ..?lrlines %lt-%lb?L/%L. .  
byte %bB?s/%s. ?e(END) :?pB%pB\%..%t
```

The prompt expansion features are also used for another purpose: if an environment variable `LESSEDIT` is defined, it is used as the command to be executed when the `v` command is invoked. The `LESSEDIT` string is expanded in the same way as the prompt strings. The default value for `LESSEDIT` is:

```
%E ?lm+%lm. %g
```

Note that this expands to the editor name, followed by a `+` and the line number, followed by the shell-escaped file name. If your editor does not accept the "`+linenumber`" syntax, or has other differences in invocation syntax, the `LESSEDIT` variable can be changed to modify this default.

SECURITY

When the environment variable `LESSSECURE` is set to 1, *less* runs in a "secure" mode. This means these features are disabled:

!	the shell command
	the pipe command
:e	the examine command.
v	the editing command
s -o	log files
-k	use of lesskey files
-t	use of tags files
	metacharacters in filenames, such as *
	filename completion (TAB, ^L)

Less can also be compiled to be permanently in "secure" mode.

COMPATIBILITY WITH MORE

If the environment variable `LESS_IS_MORE` is set to 1, or if the program is invoked via a file link named "more", *less* behaves (mostly) in conformance with the POSIX "more" command specification. In this mode, *less* behaves differently in these ways:

The `-e` option works differently. If the `-e` option is not set, *less* behaves as if the `-e` option were set. If the `-e` option is set, *less* behaves as if the `-E` option were set.

The `-m` option works differently. If the `-m` option is not set, the medium prompt is used, and it is prefixed with the string "--More--". If the `-m` option is set, the short prompt is used.

The `-n` option acts like the `-z` option. The normal behavior of the `-n` option is unavailable in this mode.

The parameter to the `-p` option is taken to be a *less* command rather than a search pattern.

The LESS environment variable is ignored, and the MORE environment variable is used in its place.

ENVIRONMENT VARIABLES

Environment variables may be specified either in the system environment as usual, or in a *lesskey*(1) file. If environment variables are defined in more than one place, variables defined in a local lesskey file take precedence over variables defined in the system environment, which take precedence over variables defined in the system-wide lesskey file.

COLUMNS

Sets the number of columns on the screen. Takes precedence over the number of columns specified by the TERM variable. (But if you have a windowing system which supports TIOCGWINSZ or WIOCGETD, the window system's idea of the screen size takes precedence over the LINES and COLUMNS environment variables.)

EDITOR

The name of the editor (used for the v command).

HOME Name of the user's home directory (used to find a lesskey file on Unix and OS/2 systems).

HOMEDRIVE, HOMEPATH

Concatenation of the HOMEDRIVE and HOMEPATH environment variables is the name of the user's home directory if the HOME variable is not set (only in the Windows version).

INIT Name of the user's init directory (used to find a lesskey file on OS/2 systems).

LANG Language for determining the character set.

LC_CTYPE

Language for determining the character set.

LESS Options which are passed to *less* automatically.

LESSANSIENDCHARS

Characters which may end an ANSI color escape sequence (default "m").

LESSANSIMIDCHARS

Characters which may appear between the ESC character and the end character in an ANSI color escape sequence (default "0123456789:[?!"'#%()^+ "].

LESSBINFMT

Format for displaying non-printable, non-control characters.

LESSCHARDEF

Defines a character set.

LESSCHARSET

Selects a predefined character set.

LESSCLOSE

Command line to invoke the (optional) input-postprocessor.

LESSECHO

Name of the lessecho program (default "lessecho"). The lessecho program is needed to expand metacharacters, such as * and ?, in filenames on Unix systems.

LESSEDIT

Editor prototype string (used for the v command). See discussion under PROMPTS.

LESSGLOBALTAGS

Name of the command used by the -t option to find global tags. Normally should be set to "global" if your system has the *global*(1) command. If not set, global tags are not used.

LESSHISTFILE

Name of the history file used to remember search commands and shell commands between invocations of *less*. If set to "-" or "/dev/null", a history file is not used. The default is "\$XDG_DATA_HOME/.lesshist" or "\$HOME/.lesshist" on Unix systems, "\$HOME/_lesshist" on DOS and Win-

dows systems, or "\$HOME/lessht.ini" or "\$INIT/lessht.ini" on OS/2 systems.

LESSHISTSIZE

The maximum number of commands to save in the history file. The default is 100.

LESSKEYIN

Name of the default *lesskey source* file.

LESSKEY

Name of the default *lesskey binary* file. (Not used if "\$LESSKEYIN" exists.)

LESSKEYIN_SYSTEM

Name of the default system-wide *lesskey source* file.

LESSKEY_SYSTEM

Name of the default system-wide *lesskey binary* file. (Not used if "\$LESSKEYIN_SYSTEM" exists.)

LESSMETACHARS

List of characters which are considered "metacharacters" by the shell.

LESSMETAESCAPE

Prefix which less will add before each metacharacter in a command sent to the shell. If LESSMETAESCAPE is an empty string, commands containing metacharacters will not be passed to the shell.

LESSOPEN

Command line to invoke the (optional) input-preprocessor.

LESSSECURE

Runs less in "secure" mode. See discussion under SECURITY.

LESSSEPARATOR

String to be appended to a directory name in filename completion.

LESSUTFBINFMT

Format for displaying non-printable Unicode code points.

LESS_IS_MORE

Emulate the *more*(1) command.

LINES Sets the number of lines on the screen. Takes precedence over the number of lines specified by the TERM variable. (But if you have a windowing system which supports TIOCGWINSZ or WIOCGETD, the window system's idea of the screen size takes precedence over the LINES and COLUMNS environment variables.)

MORE Options which are passed to *less* automatically when running in *more* compatible mode.

PATH User's search path (used to find a lesskey file on MS-DOS and OS/2 systems).

SHELL

The shell used to execute the ! command, as well as to expand filenames.

TERM The type of terminal on which *less* is being run.

VISUAL

The name of the editor (used for the v command).

SEE ALSO

lesskey(1)

COPYRIGHT

Copyright (C) 1984-2021 Mark Nudelman

less is part of the GNU project and is free software. You can redistribute it and/or modify it under the terms of either (1) the GNU General Public License as published by the Free Software Foundation; or (2) the Less License. See the file README in the less distribution for more details regarding redistribution. You

should have received a copy of the GNU General Public License along with the source for less; see the file COPYING. If not, write to the Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA. You should also have received a copy of the Less License; see the file LICENSE.

less is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

AUTHOR

Mark Nudelman

Report bugs at <https://github.com/gsws/less/issues>.

For more information, see the less homepage at

<https://greenwoodsoftware.com/less>

NAME

lessecho – expand metacharacters

SYNOPSIS

lessecho [*-ox*] [*-cx*] [*-pn*] [*-dn*] [*-mx*] [*-nn*] [*-ex*] [*-a*] *file ...*

DESCRIPTION

lessecho is a program that simply echos its arguments on standard output. But any metacharacter in the output is preceded by an "escape" character, which by default is a backslash.

OPTIONS

A summary of options is included below.

- ex** Specifies "x", rather than backslash, to be the escape char for metachars. If x is "-", no escape char is used and arguments containing metachars are surrounded by quotes instead.
- ox** Specifies "x", rather than double-quote, to be the open quote character, which is used if the -e- option is specified.
- cx** Specifies "x" to be the close quote character.
- pn** Specifies "n" to be the open quote character, as an integer.
- dn** Specifies "n" to be the close quote character, as an integer.
- mx** Specifies "x" to be a metachar. By default, no characters are considered metachars.
- nn** Specifies "n" to be a metachar, as an integer.
- fn** Specifies "n" to be the escape char for metachars, as an integer.
- a** Specifies that all arguments are to be quoted. The default is that only arguments containing metacharacters are quoted.

SEE ALSO

less(1)

AUTHOR

This manual page was written by Thomas Schoepf <schoepf@debian.org>, for the Debian GNU/Linux system (but may be used by others).

Report bugs at <https://github.com/gwsw/less/issues>.

NAME

lessfile, *lesspipe* – "input preprocessor" for *less*.

SYNOPSIS

lessfile, lesspipe

DESCRIPTION

This manual page documents briefly the *lessfile*, and *lesspipe* commands. This manual page was written for the Debian GNU/Linux distribution because the input preprocessor scripts are provided by Debian GNU/Linux and are not part of the original program.

lessfile and *lesspipe* are programs that can be used to modify the way the contents of a file are displayed in *less*. What this means is that *less* can automatically open up tar files, uncompress gzipped files, and even display something reasonable for graphics files.

lesspipe will toss the contents/info on STDOUT and *less* will read them as they come across. This means that you do not have to wait for the decoding to finish before less shows you the file. This also means that you will get a 'byte N' instead of an N% as your file position. You can seek to the end and back to get the N% but that means you have to wait for the pipe to finish.

lessfile will toss the contents/info on a file which *less* will then read. After you are done, *lessfile* will then delete the file. This means that the process has to finish before you see it, but you get nice percentages (N%) up front.

USAGE

Just put one of the following two commands in your login script (e.g. `~/.bash_profile`):

```
eval "$(lessfile)"
```

or

```
eval "$(lesspipe)"
```

FILE TYPE RECOGNITION

File types are recognized by their extensions. This is a list of currently supported extensions (grouped by the programs that handle them):

```
*.a  
*.arj  
*.tar.bz2  
*.bz  
*.bz2  
*.deb, *.udeb, *.ddeb  
*.doc  
*.egg  
*.gif, *.jpeg, *.jpg, *.pcd, *.png, *.tga, *.tiff, *.tif  
*.iso, *.raw, *.bin  
*.lha, *.lzh  
*.tar.lz, *.tlz  
*.lz  
*.7z  
*.pdf  
*.rar, *.r[0-9][0-9]  
*.rpm  
*.tar.gz, *.tgz, *.tar.z, *.tar.dz  
*.gz, *.z, *.dz  
*.tar  
*.tar.xz, *.xz  
*.whl  
*.jar, *.war, *.xpi, *.zip
```

```
*.zoo
*.tar.zst, *.tzst
*.zst
```

USER DEFINED FILTERS

It is possible to extend and overwrite the default *lesspipe* and *lessfile* input processor if you have specialized requirements. Create an executable program with the name *.lessfilter* and put it into your home directory. This can be a shell script or a binary program.

It is important that this program returns the correct exit code: return 0 if your filter handles the input, return 1 if the standard *lesspipe/lessfile* filter should handle the input.

Here is an example script:

```
#!/bin/sh

case "$1" in
  *.extension)
    extension-handler "$1"
    ;;
  *)
    # We don't handle this format.
    exit 1
esac

# No further processing by lesspipe necessary
exit 0
```

FILES

~.lessfilter

Executable file that can do user defined processing. See section USER DEFINED FILTERS for more information.

BUGS

Sometimes, less does not display the contents file you want to view but output that is produced by your login scripts (*~/.bashrc* or *~/.bash_profile*). This happens because less uses your current shell to run the *lesspipe* filter. Bash first looks for the variable *\$BASH_ENV* in the environment expands its value and uses the expanded value as the name of a file to read and execute. If this file produces any output less will display this. A way to solve this problem is to put the following lines on the top of your login script that produces output:

```
if [ -z "$PS1" ]; then
  exit
fi
```

This tests whether the prompt variable *\$PS1* is set and if it isn't (which is the case for non-interactive shells) it will exit the script.

SEE ALSO

less(1)

AUTHOR

This manual page was written by Thomas Schoepf <schoepf@debian.org>, for the Debian GNU/Linux system (but may be used by others). Most of the text was copied from a description written by Darren Stalder <torin@daft.com>.

NAME

`lesskey` – specify key bindings for `less`

SYNOPSIS (deprecated)

```
lesskey [-o output] [--] [input]
lesskey [--output=output] [--] [input]
lesskey -V
lesskey --version
```

SCOPE

This document describes the format of the `lesskey` source file, which is used by `less` version 582 and later. In previous versions of `less`, a separate program called `lesskey` was used to compile the `lesskey` source file into a format understood by `less`. This compilation step is no longer required and the `lesskey` program is therefore deprecated although the file format remains supported by `less` itself.

FILE FORMAT

The input file consists of one or more *sections*. Each section starts with a line that identifies the type of section. Possible sections are:

#command	Defines new command keys.
#line-edit	Defines new line-editing keys.
#env	Defines environment variables.

Blank lines and lines which start with a pound sign (#) are ignored, except for the special section header lines.

COMMAND SECTION

The command section begins with the line

```
#command
```

If the command section is the first section in the file, this line may be omitted. The command section consists of lines of the form:

```
string <whitespace> action [extra-string] <newline>
```

Whitespace is any sequence of one or more spaces and/or tabs. The *string* is the command key(s) which invoke the action. The *string* may be a single command key, or a sequence of up to 15 keys. The *action* is the name of the less action, from the list below. The characters in the *string* may appear literally, or be prefixed by a caret to indicate a control key. A backslash followed by one to three octal digits may be used to specify a character by its octal value. A backslash followed by certain characters specifies input characters as follows:

\b	BACKSPACE
\e	ESCAPE
\n	NEWLINE
\r	RETURN
\t	TAB
\ku	UP ARROW
\kd	DOWN ARROW
\kr	RIGHT ARROW
\kl	LEFT ARROW

\kU	PAGE UP
\kD	PAGE DOWN
\kh	HOME
\ke	END
\kx	DELETE

A backslash followed by any other character indicates that character is to be taken literally. Characters which must be preceded by backslash include caret, space, tab and the backslash itself.

An action may be followed by an "extra" string. When such a command is entered while running *less*, the action is performed, and then the extra string is parsed, just as if it were typed in to *less*. This feature can be used in certain cases to extend the functionality of a command. For example, see the "{" and ":" commands in the example below. The extra string has a special meaning for the "quit" action: when *less* quits, the first character of the extra string is used as its exit status.

EXAMPLE

The following input file describes the set of default command keys used by *less*:

```
#command
\r      forw-line
\n      forw-line
\ e    forw-line
\ j    forw-line
\kd   forw-line
^E    forw-line
^N    forw-line
\ k    back-line
\ y    back-line
^Y    back-line
^K    back-line
^P    back-line
J     forw-line-force
K     back-line-force
Y     back-line-force
d     forw-scroll
^D   forw-scroll
\ u   back-scroll
^U   back-scroll
\40  forw-screen
f     forw-screen
^F   forw-screen
^V   forw-screen
\kD  forw-screen
b     back-screen
^B   back-screen
\ev  back-screen
\kU  back-screen
z     forw-window
w     back-window
\e\40 forw-screen-force
F     forw-forever
\ef  forw-until-hilite
R     repaint-flush
```

r	repaint
^R	repaint
^L	repaint
\eu	undo-hilite
\eU	clear-search
g	goto-line
\kh	goto-line
<	goto-line
\e<	goto-line
p	percent
%	percent
\e[left-scroll
\e]	right-scroll
\e(left-scroll
\e)	right-scroll
\kl	left-scroll
\kr	right-scroll
\e{	no-scroll
\e}	end-scroll
{	forw-bracket {}
}	back-bracket {}
(forw-bracket ()
)	back-bracket ()
[forw-bracket []
]	back-bracket []
\eF	forw-bracket
\e^B	back-bracket
G	goto-end
\e>	goto-end
>	goto-end
\ke	goto-end
\eG	goto-end-buffered
=	status
^G	status
:f	status
/	forw-search
?	back-search
\e/	forw-search *
\e?	back-search *
n	repeat-search
\en	repeat-search-all
N	reverse-search
\eN	reverse-search-all
&	filter
m	set-mark
M	set-mark-bottom
\em	clear-mark
,	goto-mark
^X^X	goto-mark
E	examine
:e	examine
^X^V	examine
:n	next-file

:p	prev-file
t	next-tag
T	prev-tag
:x	index-file
:d	remove-file
-	toggle-option
:t	toggle-option t
s	toggle-option o
-	display-option
	pipe
v	visual
!	shell
+	firstcmd
H	help
h	help
V	version
0	digit
1	digit
2	digit
3	digit
4	digit
5	digit
6	digit
7	digit
8	digit
9	digit
q	quit
Q	quit
:q	quit
:Q	quit
ZZ	quit

PRECEDENCE

Commands specified by *lesskey* take precedence over the default commands. A default command key may be disabled by including it in the input file with the action "invalid". Alternatively, a key may be defined to do nothing by using the action "noaction". "noaction" is similar to "invalid", but *less* will give an error beep for an "invalid" command, but not for a "noaction" command. In addition, ALL default commands may be disabled by adding this control line to the input file:

```
#stop
```

This will cause all default commands to be ignored. The #stop line should be the last line in that section of the file.

Be aware that #stop can be dangerous. Since all default commands are disabled, you must provide sufficient commands before the #stop line to enable all necessary actions. For example, failure to provide a "quit" command can lead to frustration.

LINE EDITING SECTION

The line-editing section begins with the line:

```
#line-edit
```

This section specifies new key bindings for the line editing commands, in a manner similar to the way key bindings for ordinary commands are specified in the #command section. The line-editing section consists

of a list of keys and actions, one per line as in the example below.

EXAMPLE

The following input file describes the set of default line-editing keys used by less:

```
#line-edit
\t      forw-complete
\17    back-complete
\e\t   back-complete
^L     expand
^V     literal
^A     literal
\el    right
\kr    right
\eh    left
\kl    left
\eb    word-left
\ekl   word-left
\ew    word-right
\ekr   word-right
\ei    insert
\ex    delete
\kx    delete
\ex    word-delete
\ekx   word-delete
\eb    word-backspace
\eo    home
\kh    home
\es    end
\ke    end
\ek    up
\ku    up
\ej    down
^G    abort
```

LESS ENVIRONMENT VARIABLES

The environment variable section begins with the line

```
#env
```

Following this line is a list of environment variable assignments. Each line consists of an environment variable name, an equals sign (=) and the value to be assigned to the environment variable. White space before and after the equals sign is ignored. Variables assigned in this way are visible only to *less*. If a variable is specified in the system environment and also in a lesskey file, the value in the lesskey file takes precedence. Although the lesskey file can be used to override variables set in the environment, the main purpose of assigning variables in the lesskey file is simply to have all *less* configuration information stored in one file.

EXAMPLE

The following input file sets the *-i* option whenever *less* is run, and specifies the character set to be "latin1":

```
#env
LESS = -i
LESSCHARSET = latin1
```

SEE ALSO

less(1)

WARNINGS

On MS-DOS and OS/2 systems, certain keys send a sequence of characters which start with a NUL character (0). This NUL character should be represented as \340 in a lesskey file.

COPYRIGHT

Copyright (C) 1984-2021 Mark Nudelman

less is part of the GNU project and is free software. You can redistribute it and/or modify it under the terms of either (1) the GNU General Public License as published by the Free Software Foundation; or (2) the Less License. See the file README in the less distribution for more details regarding redistribution. You should have received a copy of the GNU General Public License along with the source for less; see the file COPYING. If not, write to the Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA. You should also have received a copy of the Less License; see the file LICENSE.

less is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

AUTHOR

Mark Nudelman

Report bugs at <https://github.com/gswsw/less/issues>.

NAME

lessfile, *lesspipe* – "input preprocessor" for *less*.

SYNOPSIS

lessfile, lesspipe

DESCRIPTION

This manual page documents briefly the *lessfile*, and *lesspipe* commands. This manual page was written for the Debian GNU/Linux distribution because the input preprocessor scripts are provided by Debian GNU/Linux and are not part of the original program.

lessfile and *lesspipe* are programs that can be used to modify the way the contents of a file are displayed in *less*. What this means is that *less* can automatically open up tar files, uncompress gzipped files, and even display something reasonable for graphics files.

lesspipe will toss the contents/info on STDOUT and *less* will read them as they come across. This means that you do not have to wait for the decoding to finish before less shows you the file. This also means that you will get a 'byte N' instead of an N% as your file position. You can seek to the end and back to get the N% but that means you have to wait for the pipe to finish.

lessfile will toss the contents/info on a file which *less* will then read. After you are done, *lessfile* will then delete the file. This means that the process has to finish before you see it, but you get nice percentages (N%) up front.

USAGE

Just put one of the following two commands in your login script (e.g. `~/.bash_profile`):

```
eval "$(lessfile)"
```

or

```
eval "$(lesspipe)"
```

FILE TYPE RECOGNITION

File types are recognized by their extensions. This is a list of currently supported extensions (grouped by the programs that handle them):

```
*.a  
*.arj  
*.tar.bz2  
*.bz  
*.bz2  
*.deb, *.udeb, *.ddeb  
*.doc  
*.egg  
*.gif, *.jpeg, *.jpg, *.pcd, *.png, *.tga, *.tiff, *.tif  
*.iso, *.raw, *.bin  
*.lha, *.lzh  
*.tar.lz, *.tlz  
*.lz  
*.7z  
*.pdf  
*.rar, *.r[0-9][0-9]  
*.rpm  
*.tar.gz, *.tgz, *.tar.z, *.tar.dz  
*.gz, *.z, *.dz  
*.tar  
*.tar.xz, *.xz  
*.whl  
*.jar, *.war, *.xpi, *.zip
```

```
*.zoo
*.tar.zst, *.tzst
*.zst
```

USER DEFINED FILTERS

It is possible to extend and overwrite the default *lesspipe* and *lessfile* input processor if you have specialized requirements. Create an executable program with the name *.lessfilter* and put it into your home directory. This can be a shell script or a binary program.

It is important that this program returns the correct exit code: return 0 if your filter handles the input, return 1 if the standard *lesspipe/lessfile* filter should handle the input.

Here is an example script:

```
#!/bin/sh

case "$1" in
  *.extension)
    extension-handler "$1"
    ;;
  *)
    # We don't handle this format.
    exit 1
esac

# No further processing by lesspipe necessary
exit 0
```

FILES

~.lessfilter

Executable file that can do user defined processing. See section USER DEFINED FILTERS for more information.

BUGS

Sometimes, less does not display the contents file you want to view but output that is produced by your login scripts (*~/.bashrc* or *~/.bash_profile*). This happens because less uses your current shell to run the *lesspipe* filter. Bash first looks for the variable *\$BASH_ENV* in the environment expands its value and uses the expanded value as the name of a file to read and execute. If this file produces any output less will display this. A way to solve this problem is to put the following lines on the top of your login script that produces output:

```
if [ -z "$PS1" ]; then
  exit
fi
```

This tests whether the prompt variable *\$PS1* is set and if it isn't (which is the case for non-interactive shells) it will exit the script.

SEE ALSO

less(1)

AUTHOR

This manual page was written by Thomas Schoepf <schoepf@debian.org>, for the Debian GNU/Linux system (but may be used by others). Most of the text was copied from a description written by Darren Stalder <torin@daft.com>.

NAME

plocate – find files by name, quickly

SYNOPSIS

plocate [*OPTION*]... *PATTERN*...

DESCRIPTION

plocate finds all files on the system matching the given pattern (or all of the patterns if multiple are given). It does this by means of an index made by **updatedb(8)** or (less commonly) converted from another index by **plocate-build(8)**.

plocate is largely argument-compatible with **mlocate(1)**, but is significantly faster. In particular, it rarely needs to scan through its entire database, unless the pattern is very short (less than three bytes) or you want to search for a regular expression. It does not try to maintain compatibility with BSD locate, or non-UTF-8 filenames and locales. Most I/O is done asynchronously, but the results are synchronized so that output comes in the same order every time.

When multiple patterns are given, **plocate** will search for files that match *all* of them. This is the main incompatibility with **mlocate(1)**, which searches for files that match one or more patterns, unless the **-A** option is given.

By default, patterns are taken to be substrings to search for. If at least one non-escaped globbing metacharacter (*, ?, or []) is given, that pattern is instead taken to be a glob pattern (which means it needs to start and end in * for a substring match). If **--regexp** is given, patterns are instead taken to be (non-anchored) POSIX basic regular expressions, and if **--regex** is given, patterns are taken to be POSIX extended regular expressions. All of this matches **mlocate(1)** behavior.

Like **mlocate(1)**, **plocate** shows all files visible to the calling user (by virtue of having read and execute permissions on all parent directories), and none that are not, by means of running with the setgid bit set to access the index (which is built as root), but by testing visibility as the calling user.

OPTIONS**-A, --all**

Ignored for compatibility with **mlocate(1)**.

-b, --basename

Match only against the file name portion of the path name, ie., the directory names will be excluded from the match (but still printed). This does not speed up the search, but can suppress uninteresting matches.

-c, --count

Do not print each match. Instead, count them, and print out a total number at the end.

-d, --database DBPATH

Find matches in the given database, instead of **/var/lib/plocate/plocate.db**. This argument can be given multiple times, to search multiple databases. It is also possible to give multiple databases in one argument, separated by :. (Any character, including :, and \, can be escaped by prepending a \.)

-e, --existing

Print only entries that refer to files existing at the time **locate** is run. Note that unlike **mlocate(1)**, symlinks are not followed by default (and indeed, there is no option to change this).

-i, --ignore-case

Do a case-insensitive match as given by the current locale (default is case-sensitive, byte-by-byte match). Note that **plocate** does not support the full range of Unicode case folding rules; in particular, searching for β will not give you matches on ss even in a German locale. Also note that this option will be somewhat slower than a case-sensitive match, since it needs to generate more candidates for searching the index.

-p, --ignore-spaces

Ignore punctuation and spaces when matching patterns.

-l, --limit LIMIT

Stop searching after *LIMIT* matches have been found. If **--count** is given, the number printed out will be at most *LIMIT*.

-N, --literal

Print entry names without quoting. Normally, **plocate** will escape special characters in filenames, so that they are safe for consumption by typical shells (similar to the GNU coreutils *shell-escape-always* quoting style), unless printing to a pipe, but this option will turn off such quoting.

-0, --null

Instead of writing a newline after every match, write a NUL (ASCII 0). This is useful for creating unambiguous output when it is to be processed by other tools (like **xargs(1)**), as filenames are allowed to contain embedded newlines.

-r, --regexp

Patterns are taken to be POSIX basic regular expressions. See **egrep(7)** for more information. Note that this forces a linear scan through the entire database, which is slow.

--regex

Like **--regexp**, but patterns are instead taken to be POSIX *extended* regular expressions.

-w, --wholename

Match against the entire path name. This is the default, so unless **-b** is given first (see above), it will not do anything. This option thus exists only as compatibility with **mlocate(1)**.

--help Print out usage information, then exit successfully.**--version**

Print out version information, then exit successfully.

ENVIRONMENT**LOCATE_PATH**

If given, appended after the list of **--database** paths (whether an explicit is given or the default is used). Colon-delimiting and character escaping follows the same rules as for **--database**.

AUTHOR

Steinar H. Gunderson <steinar+plocate@gunderson.no>

SEE ALSO

plocate-build(8), mlocate(1), updatedb(8)

NAME

ls – list directory contents

SYNOPSIS

ls [*OPTION*]... [*FILE*]...

DESCRIPTION

List information about the **FILEs** (the current directory by default). Sort entries alphabetically if none of **-cftuvSUX** nor **--sort** is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with `.`

-A, --almost-all

do not list implied `.` and `..`

--author

with **-l**, print the author of each file

-b, --escape

print C-style escapes for nongraphic characters

--block-size=SIZE

with **-l**, scale sizes by **SIZE** when printing them; e.g., '**--block-size=M**'; see **SIZE** format below

-B, --ignore-backups

do not list implied entries ending with `~`

-c with **-lt**: sort by, and show, ctime (time of last modification of file status information); with **-l**: show ctime and sort by name; otherwise: sort by ctime, newest first

-C list entries by columns

--color[=WHEN]

colorize the output; **WHEN** can be 'always' (default if omitted), 'auto', or 'never'; more info below

-d, --directory

list directories themselves, not their contents

-D, --dirend

generate output designed for Emacs' **dirend** mode

-f do not sort, enable **-aU**, disable **-ls --color**

-F, --classify

append indicator (one of `*/=>@|`) to entries

--file-type

likewise, except do not append `*`

--format=WORD

across **-x**, commas **-m**, horizontal **-x**, long **-l**, single-column **-1**, verbose **-l**, vertical **-C**

--full-time

like **-l --time-style=full-iso**

-g like **-l**, but do not list owner

--group-directories-first

group directories before files;

can be augmented with a **--sort** option, but any use of **--sort=none** (**-U**) disables grouping

-G, --no-group
 in a long listing, don't print group names

-h, --human-readable
 with -l and -s, print sizes like 1K 234M 2G etc.

--si likewise, but use powers of 1000 not 1024

-H, --dereference-command-line
 follow symbolic links listed on the command line

--dereference-command-line-symlink-to-dir
 follow each command line symbolic link
 that points to a directory

--hide= PATTERN
 do not list implied entries matching shell PATTERN (overridden by -a or -A)

--hyperlink[= WHEN]
 hyperlink file names; WHEN can be 'always' (default if omitted), 'auto', or 'never'

--indicator-style=WORD
 append indicator with style WORD to entry names: none (default), slash (-p), file-type
 (--file-type), classify (-F)

-i, --inode
 print the index number of each file

-I, --ignore= PATTERN
 do not list implied entries matching shell PATTERN

-k, --kibibytes
 default to 1024-byte blocks for disk usage; used only with -s and per directory totals

-l use a long listing format

-L, --dereference
 when showing file information for a symbolic link, show information for the file the link references rather than for the link itself

-m fill width with a comma separated list of entries

-n, --numeric-uid-gid
 like -l, but list numeric user and group IDs

-N, --literal
 print entry names without quoting

-o like -l, but do not list group information

-p, --indicator-style=slash
 append / indicator to directories

-q, --hide-control-chars
 print ? instead of nongraphic characters

--show-control-chars
 show nongraphic characters as-is (the default, unless program is 'ls' and output is a terminal)

-Q, --quote-name
 enclose entry names in double quotes

--quoting-style=WORD
 use quoting style WORD for entry names: literal, locale, shell, shell-always, shell-escape,
 shell-escape-always, c, escape (overrides QUOTING_STYLE environment variable)

-r, --reverse
 reverse order while sorting

-R, --recursive
 list subdirectories recursively

-s, --size
 print the allocated size of each file, in blocks

-S sort by file size, largest first

--sort=WORD
 sort by WORD instead of name: none (-U), size (-S), time (-t), version (-v), extension (-X)

--time=WORD
 change the default of using modification times; access time (-u): atime, access, use; change time (-c): ctime, status; birth time: birth, creation;
 with -l, WORD determines which time to show; with --sort=time, sort by WORD (newest first)

--time-style=TIME_STYLE
 time/date format with -l; see TIME_STYLE below

-t sort by time, newest first; see --time

-T, --tabsize=COLS
 assume tab stops at each COLS instead of 8

-u with -lt: sort by, and show, access time; with -l: show access time and sort by name; otherwise:
 sort by access time, newest first

-U do not sort; list entries in directory order

-v natural sort of (version) numbers within text

-w, --width=COLS
 set output width to COLS. 0 means no limit

-x list entries by lines instead of by columns

-X sort alphabetically by entry extension

-Z, --context
 print any security context of each file

-1 list one file per line. Avoid '\n' with -q or -b

--help display this help and exit

--version
 output version information and exit

The SIZE argument is an integer and optional unit (example: 10K is 10*1024). Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000). Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

The TIME_STYLE argument can be full-iso, long-iso, iso, locale, or +FORMAT. FORMAT is interpreted like in date(1). If FORMAT is FORMAT1<newline>FORMAT2, then FORMAT1 applies to non-recent files and FORMAT2 to recent files. TIME_STYLE prefixed with 'posix-' takes effect only outside the POSIX locale. Also the TIME_STYLE environment variable sets the default style to use.

Using color to distinguish file types is disabled both by default and with --color=never. With --color=auto, ls emits color codes only when standard output is connected to a terminal. The LS_COLORS environment variable can change the settings. Use the dircolors command to set it.

Exit status:

0 if OK,

- 1 if minor problems (e.g., cannot access subdirectory),
- 2 if serious trouble (e.g., cannot access command-line argument).

AUTHOR

Written by Richard M. Stallman and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent
permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/ls>>

or available locally via: info '(coreutils) ls invocation'

NAME

lsblk – list block devices

SYNOPSIS

lsblk [options] [*device...*]

DESCRIPTION

lsblk lists information about all available or the specified block devices. The **lsblk** command reads the **sysfs** filesystem and **udev db** to gather information. If the udev db is not available or **lsblk** is compiled without udev support, then it tries to read LABELs, UUIDs and filesystem types from the block device. In this case root permissions are necessary.

The command prints all block devices (except RAM disks) in a tree-like format by default. Use **lsblk --help** to get a list of all available columns.

The default output, as well as the default output from options like **--fs** and **--topology**, is subject to change. So whenever possible, you should avoid using default outputs in your scripts. Always explicitly define expected columns by using **--output columns-list** and **--list** in environments where a stable output is required.

Note that **lsblk** might be executed in time when **udev** does not have all information about recently added or modified devices yet. In this case it is recommended to use **udevadm settle** before **lsblk** to synchronize with udev.

The relationship between block devices and filesystems is not always one-to-one. The filesystem may use more block devices, or the same filesystem may be accessible by more paths. This is the reason why **lsblk** provides MOUNTPOINT and MOUNTPOINTS (pl.) columns. The column MOUNTPOINT displays only one mount point (usually the last mounted instance of the filesystem), and the column MOUNTPOINTS displays by multi-line cell all mount points associated with the device.

OPTIONS

-A, --noempty

Don't print empty devices.

-a, --all

Disable all built-in filters and list all empty devices and RAM disk devices too.

-b, --bytes

Print the sizes in bytes rather than in a human-readable format.

By default, the unit, sizes are expressed in, is byte, and unit prefixes are in power of 2^10 (1024).

Abbreviations of symbols are exhibited truncated in order to reach a better readability, by exhibiting alone the first letter of them; examples: "1 KiB" and "1 MiB" are respectively exhibited as "1 K" and "1 M", then omitting on purpose the mention "iB", which is part of these abbreviations.

-D, --discard

Print information about the discarding capabilities (TRIM, UNMAP) for each device.

-d, --nodeps

Do not print holder devices or slaves. For example, **lsblk --nodeps /dev/sda** prints information about the sda device only.

-E, --dedup column

Use *column* as a de-duplication key to de-duplicate output tree. If the key is not available for the device, or the device is a partition and parental whole-disk device provides the same key than the device is always printed.

The usual use case is to de-duplicate output on system multi-path devices, for example by **-E WWN**.

-e, --exclude *list*

Exclude the devices specified by the comma-separated *list* of major device numbers. Note that RAM disks (major=1) are excluded by default if **--all** is not specified. The filter is applied to the top-level devices only. This may be confusing for **--list** output format where hierarchy of the devices is not obvious.

-f, --fs

Output info about filesystems. This option is equivalent to **-o NAME,FSTYPE,FSVER,LABEL,UUID,FSAVAIL,FSUSE%,MOUNTPOINTS**. The authoritative information about filesystems and raids is provided by the **blkid(8)** command.

-I, --include *list*

Include devices specified by the comma-separated *list* of major device numbers. The filter is applied to the top-level devices only. This may be confusing for **--list** output format where hierarchy of the devices is not obvious.

-i, --ascii

Use ASCII characters for tree formatting.

-J, --json

Use JSON output format. It's strongly recommended to use **--output** and also **--tree** if necessary.

-l, --list

Produce output in the form of a list. The output does not provide information about relationships between devices and since version 2.34 every device is printed only once if **--pairs** or **--raw** not specified (the parsable outputs are maintained in backwardly compatible way).

-M, --merge

Group parents of sub-trees to provide more readable output for RAIDs and Multi-path devices. The tree-like output is required.

-m, --perms

Output info about device owner, group and mode. This option is equivalent to **-o NAME,SIZE,OWNER,GROUP,MODE**.

-n, --noheadings

Do not print a header line.

-o, --output *list*

Specify which output columns to print. Use **--help** to get a list of all supported columns. The columns may affect tree-like output. The default is to use tree for the column 'NAME' (see also **--tree**).

The default list of columns may be extended if *list* is specified in the format **+list** (e.g., **lsblk -o +UUID**).

-O, --output-all

Output all available columns.

-P, --pairs

Produce output in the form of key="value" pairs. The output lines are still ordered by dependencies. All potentially unsafe value characters are hex-escaped (`\x<code>`). See also option **--shell**.

-p, --paths

Print full device paths.

-r, --raw

Produce output in raw format. The output lines are still ordered by dependencies. All potentially unsafe characters are hex-escaped (`\x<code>`) in the NAME, KNAME, LABEL, PARTLABEL and MOUNTPOINT columns.

-S, --scsi

Output info about SCSI devices only. All partitions, slaves and holder devices are ignored.

-s, --inverse

Print dependencies in inverse order. If the **--list** output is requested then the lines are still ordered by dependencies.

-T, --tree[=column]

Force tree-like output format. If *column* is specified, then a tree is printed in the column. The default is NAME column.

-t, --topology

Output info about block-device topology. This option is equivalent to

-o

NAME,ALIGNMENT,MIN-IO,OPT-IO,PHY-SEC,LOG-SEC,ROTA,SCHED,RQ-SIZE,RA,WSAME.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

-w, --width number

Specifies output width as a number of characters. The default is the number of the terminal columns, and if not executed on a terminal, then output width is not restricted at all by default. This option also forces **lsblk** to assume that terminal control characters and unsafe characters are not allowed. The expected use-case is for example when **lsblk** is used by the **watch(1)** command.

-x, --sort column

Sort output lines by *column*. This option enables **--list** output format by default. It is possible to use the option **--tree** to force tree-like output and than the tree branches are sorted by the *column*.

-y, --shell

The column name will be modified to contain only characters allowed for shell variable identifiers, for example, MIN_IO and FSUSE_PCT instead of MIN-IO and FSUSE%. This is usable, for example, with **--pairs**. Note that this feature has been automatically enabled for **--pairs** in version 2.37, but due to compatibility issues, now it's necessary to request this behavior by **--shell**.

-z, --zoned

Print the zone related information for each device.

--sysroot directory

Gather data for a Linux instance other than the instance from which the **lsblk** command is issued. The specified directory is the system root of the Linux instance to be inspected. The real device nodes in the target directory can be replaced by text files with udev attributes.

EXIT STATUS

0	success
1	failure
32	none of specified devices found
64	some specified devices found, some not found

ENVIRONMENT

LSBLK_DEBUG=all
enables **lsblk** debug output.

LIBBLKID_DEBUG=all
enables **libblkid** debug output.

LIBMOUNT_DEBUG=all
enables **libmount** debug output.

LIBSMARTCOLS_DEBUG=all
enables **libsmartcols** debug output.

LIBSMARTCOLS_DEBUG_PADDING=on
use visible padding characters.

NOTES

For partitions, some information (e.g., queue attributes) is inherited from the parent device.

The **lsblk** command needs to be able to look up each block device by major:minor numbers, which is done by using `/sys/dev/block`. This sysfs block directory appeared in kernel 2.6.27 (October 2008). In case of problems with a new enough kernel, check that **CONFIG_SYSFS** was enabled at the time of the kernel build.

AUTHORS

Milan Broz <mbroz@redhat.com>, Karel Zak <kzak@redhat.com>

SEE ALSO

ls(1), blkid(8), findmnt(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **lsblk** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

lscpu – display information about the CPU architecture

SYNOPSIS

lscpu [options]

DESCRIPTION

lscpu gathers CPU architecture information from *sysfs*, */proc/cpuinfo* and any applicable architecture-specific libraries (e.g. **librtas** on Powerpc). The command output can be optimized for parsing or for easy readability by humans. The information includes, for example, the number of CPUs, threads, cores, sockets, and Non-Uniform Memory Access (NUMA) nodes. There is also information about the CPU caches and cache sharing, family, model, bogoMIPS, byte order, and stepping.

The default output formatting on terminal is subject to change and maybe optimized for better readability. The output for non-terminals (e.g., pipes) is never affected by this optimization and it is always in "Field: data\n" format. Use for example "**lscpu | less**" to see the default output without optimizations.

In virtualized environments, the CPU architecture information displayed reflects the configuration of the guest operating system which is typically different from the physical (host) system. On architectures that support retrieving physical topology information, **lscpu** also displays the number of physical sockets, chips, cores in the host system.

Options that result in an output table have a *list* argument. Use this argument to customize the command output. Specify a comma-separated list of column labels to limit the output table to only the specified columns, arranged in the specified order. See **COLUMNS** for a list of valid column labels. The column labels are not case sensitive.

Not all columns are supported on all architectures. If an unsupported column is specified, **lscpu** prints the column but does not provide any data for it.

The cache sizes are reported as summary from all CPUs. The versions before v2.34 reported per-core sizes, but this output was confusing due to complicated CPUs topology and the way how caches are shared between CPUs. For more details about caches see **--cache**. Since version v2.37 **lscpu** follows cache IDs as provided by Linux kernel and it does not always start from zero.

OPTIONS

-a, --all

Include lines for online and offline CPUs in the output (default for **-e**). This option may only be specified together with option **-e** or **-p**.

-B, --bytes

Print the sizes in bytes rather than in a human-readable format.

By default, the unit, sizes are expressed in, is byte, and unit prefixes are in power of 2^{10} (1024). Abbreviations of symbols are exhibited truncated in order to reach a better readability, by exhibiting alone the first letter of them; examples: "1 KiB" and "1 MiB" are respectively exhibited as "1 K" and "1 M", then omitting on purpose the mention "iB", which is part of these abbreviations.

-b, --online

Limit the output to online CPUs (default for **-p**). This option may only be specified together with option **-e** or **-p**.

-C, --caches[=list]

Display details about CPU caches. For details about available information see **--help** output.

If the *list* argument is omitted, all columns for which data is available are included in the command

output.

When specifying the *list* argument, the string of option, equal sign (=), and *list* must not contain any blanks or other whitespace. Examples: **-C=NAME,ONE-SIZE** or **--caches=NAME,ONE-SIZE**.

The default list of columns may be extended if list is specified in the format +list (e.g., **lscpu -C=+ALLOC-POLICY**).

-c, --offline

Limit the output to offline CPUs. This option may only be specified together with option **-e** or **-p**.

-e, --extended[=list]

Display the CPU information in human-readable format.

If the *list* argument is omitted, the default columns are included in the command output. The default output is subject to change.

When specifying the *list* argument, the string of option, equal sign (=), and *list* must not contain any blanks or other whitespace. Examples: **'-e=cpu,node'** or **'--extended=cpu,node'**.

The default list of columns may be extended if list is specified in the format +list (e.g., **lscpu -e=+MHZ**).

-J, --json

Use JSON output format for the default summary or extended output (see **--extended**).

-p, --parse[=list]

Optimize the command output for easy parsing.

If the *list* argument is omitted, the command output is compatible with earlier versions of **lscpu**. In this compatible format, two commas are used to separate CPU cache columns. If no CPU caches are identified the cache column is omitted. If the *list* argument is used, cache columns are separated with a colon (:).

When specifying the *list* argument, the string of option, equal sign (=), and *list* must not contain any blanks or other whitespace. Examples: **'-p=cpu,node'** or **'--parse=cpu,node'**.

The default list of columns may be extended if list is specified in the format +list (e.g., **lscpu -p=+MHZ**).

-s, --sysroot directory

Gather CPU data for a Linux instance other than the instance from which the **lscpu** command is issued. The specified *directory* is the system root of the Linux instance to be inspected.

-x, --hex

Use hexadecimal masks for CPU sets (for example "ff"). The default is to print the sets in list format (for example 0,1). Note that before version 2.30 the mask has been printed with 0x prefix.

-y, --physical

Display physical IDs for all columns with topology elements (core, socket, etc.). Other than logical IDs, which are assigned by **lscpu**, physical IDs are platform-specific values that are provided by the kernel. Physical IDs are not necessarily unique and they might not be arranged sequentially. If the kernel could not retrieve a physical ID for an element **lscpu** prints the dash (-) character.

The CPU logical numbers are not affected by this option.

--output-all

Output all available columns. This option must be combined with either **--extended**, **--parse** or **--caches**.

BUGS

The basic overview of CPU family, model, etc. is always based on the first CPU only.

Sometimes in Xen Dom0 the kernel reports wrong data.

On virtual hardware the number of cores per socket, etc. can be wrong.

AUTHORS

[Cai Qian](#) <qcai@redhat.com>, [Karel Zak](#) <kzak@redhat.com>, [Heiko Carstens](#) <heiko.carstens@de.ibm.com>

SEE ALSO

[chcpu\(8\)](#)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **lscpu** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](#) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

lshw – list hardware

SYNOPSIS

lshw [**-version**]

lshw [**-help**]

lshw [**-X**]

lshw [[**-html**] [**-short**] [**-xml**] [**-json**] [**-businfo**]] [**-dump** *filename*] [**-class** *class...*] [**-disable** *test...*] [**-enable** *test...*] [**-sanitize**] [**-numeric**] [**-quiet**] [**-notime**]

DESCRIPTION

lshw is a small tool to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc. on DMI-capable x86 or IA-64 systems and on some PowerPC machines (PowerMac G4 is known to work).

It currently supports DMI (x86 and IA-64 only), OpenFirmware device tree (PowerPC only), PCI/AGP, CPUID (x86), IDE/ATA/ATAPI, PCMCIA (only tested on x86), SCSI and USB.

-version

Displays the version of **lshw** and exits.

-help

Displays the available command line options and quits.

-X

Launch the X11 GUI (if available).

-html

Outputs the device tree as an HTML page.

-xml

Outputs the device tree as an XML tree.

-json

Outputs the device tree as a JSON object (JavaScript Object Notation).

-short

Outputs the device tree showing hardware paths, very much like the output of HP-UX's **ioscan**.

-businfo

Outputs the device list showing bus information, detailing SCSI, USB, IDE and PCI addresses.

-dump *filename*

Display output and dump collected information into a file (SQLite database).

-class *class*

Only show the given class of hardware. *class* can be found using **lshw -short** or **lshw -businfo**.

-C *class*

Alias for **-class** *class*.

-enable *test***-disable** *test*

Enables or disables a test. *test* can be **dmi** (for DMI/SMBIOS extensions), **device-tree** (for OpenFirmware device tree), **spd** (for memory Serial Presence Detect), **memory** (for memory-size guessing heuristics), **cpuinfo** (for kernel-reported CPU detection), **cpuid** (for CPU detection), **pci** (for PCI/AGP access), **isapnp** (for ISA PnP extensions), **pcmcia** (for PCMCIA/PCCARD), **ide** (for IDE/ATAPI), **usb** (for USB devices), **scsi** (for SCSI) or **network** (for network interfaces detection).

-quiet

Don't display status.

-sanitize

Remove potentially sensitive information from output (IP addresses, serial numbers, etc.).

-numeric

Also display numeric IDs (for PCI and USB devices).

-notime

Exclude volatile attributes (timestamps) from output.

BUGS

lshw currently does not detect Firewire(IEEE1394) devices.

Not all architectures supported by GNU/Linux are fully supported (e.g. CPU detection).

"Virtual" SCSI interfaces used for SCSI emulation over IDE are not reported correctly yet.

NOTES

lshw must be run as super user or it will only report partial information.

FILES

/usr/local/share/pci.ids

/usr/share/pci.ids

/etc/pci.ids

/usr/share/hwdata/pci.ids

A list of all known PCI ID's (vendors, devices, classes and subclasses). If compiled with zlib support, **lshw** will look for *pci.ids.gz* first, then for *pci.ids*.

/proc/bus/pci/*

Used to access the configuration of installed PCI buses and devices.

/proc/ide/*

Used to access the configuration of installed IDE buses and devices.

/proc/scsi/*, /dev/sg*

Used to access the configuration of installed SCSI devices.

/dev/cpu/*/cpuid

Used on x86 platforms to access CPU-specific configuration.

/proc/device-tree/*

Used on PowerPC platforms to access OpenFirmware configuration.

/proc/bus/usb/*

Used to access the configuration of installed USB buses and devices.

/sys/* Used on 2.6 kernels to access hardware/driver configuration information.

EXAMPLES

lshw -short

Lists hardware in a compact format.

lshw -class disk -class storage

Lists all disks and storage controllers in the system.

lshw -html -class network

Lists all network interfaces in HTML.

lshw -disable dmi

Don't use DMI to detect hardware.

SEE ALSO

*/proc/**, **linuxinfo(1)**, **lspci(8)**, **lsusb(8)**

COPYING

lshw is distributed under the GNU GENERAL PUBLIC LICENSE (GPL) version 2.

AUTHOR

lshw is maintained by Lyonel Vincent <lyonel@ezix.org>.

OTHER INFO

The webpage for **lshw** is at
<URL:<http://lshw.ezix.org/>>

NAME

lsmod – Show the status of modules in the Linux Kernel

SYNOPSIS

lsmod

DESCRIPTION

lsmod is a trivial program which nicely formats the contents of the /proc/modules, showing what kernel modules are currently loaded.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

insmod(8), **modprobe(8)**, **modinfo(8)** **depmod(8)**

AUTHORS

Jon Masters <jcm@jonmasters.org>

Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>

Developer

NAME

lsof – list open files

SYNOPSIS

```
lsof [ -?abChlnNOPRtUvVX ] [ -A A ] [ -c c ] [ +c c ] [ +|-d d ] [ +|-D D ] [ +|-e s ] [ +|-E ] [ +|-f
[cfgGn] ] [ -F [f] ] [ -g [s] ] [ -i [i] ] [ -k k ] [ -K k ] [ +|-L [l] ] [ +|-m m ] [ +|-M ] [ -o [o] ] [ -p s ] [
+|-r [t[m<fmt>]] ] [ -s [p:s] ] [ -S [t] ] [ -T [t] ] [ -u s ] [ +|-w ] [ -x [fl] ] [ -z [z] ] [ -Z [Z] ] [ -- ]
[names]
```

DESCRIPTION

Lsof revision 4.93.2 lists on its standard output file information about files opened by processes for the following UNIX dialects:

Apple Darwin 9 and Mac OS X 10.[567]

FreeBSD 8.[234], 9.0 and 1[012].0 for AMD64-based systems

Linux 2.1.72 and above for x86-based systems

Solaris 9, 10 and 11

(See the **DISTRIBUTION** section of this manual page for information on how to obtain the latest *lsof* revision.)

An open file may be a regular file, a directory, a block special file, a character special file, an executing text reference, a library, a stream or a network file (Internet socket, NFS file or UNIX domain socket.) A specific file or all the files in a file system may be selected by path.

Instead of a formatted display, *lsof* will produce output that can be parsed by other programs. See the **-F** option description, and the **OUTPUT FOR OTHER PROGRAMS** section for more information.

In addition to producing a single output list, *lsof* will run in repeat mode. In repeat mode it will produce output, delay, then repeat the output operation until stopped with an interrupt or quit signal. See the **+|-r [t[m<fmt>]]** option description for more information.

OPTIONS

In the absence of any options, *lsof* lists all open files belonging to all active processes.

If any list request option is specified, other list requests must be specifically requested – e.g., if **-U** is specified for the listing of UNIX socket files, NFS files won't be listed unless **-N** is also specified; or if a user list is specified with the **-u** option, UNIX domain socket files, belonging to users not in the list, won't be listed unless the **-U** option is also specified.

Normally list options that are specifically stated are ORed – i.e., specifying the **-i** option without an address and the **-ufoo** option produces a listing of all network files OR files belonging to processes owned by user “foo”. The exceptions are:

- 1) the ‘^’ (negated) login name or user ID (UID), specified with the **-u** option;
- 2) the ‘^’ (negated) process ID (PID), specified with the **-p** option;
- 3) the ‘^’ (negated) process group ID (PGID), specified with the **-g** option;
- 4) the ‘^’ (negated) command, specified with the **-c** option;
- 5) the (^) negated TCP or UDP protocol state names, specified with the **-s [p:s]** option.

Since they represent exclusions, they are applied without ORing or ANDing and take effect before any other selection criteria are applied.

The **-a** option may be used to AND the selections. For example, specifying **-a**, **-U**, and **-ufoo** produces a listing of only UNIX socket files that belong to processes owned by user “foo”.

Caution: the **-a** option causes all list selection options to be ANDed; it can't be used to cause ANDing of selected pairs of selection options by placing it between them, even though its placement there is acceptable. Wherever **-a** is placed, it causes the ANDing of all selection options.

Items of the same selection set – command names, file descriptors, network addresses, process identifiers, user identifiers, zone names, security contexts – are joined in a single ORed set and applied before the result participates in ANDing. Thus, for example, specifying **-i@aaa.bbb**, **-i@ccc.ddd**, **-a**, and **-u^{ffff}.ggg** will select the listing of files that belong to either login “fff” OR “ggg” AND have network connections to either host aaa.bbb OR ccc.ddd.

Options may be grouped together following a single prefix -- e.g., the option set “**-a -b -C**” may be stated as **-abC**. However, since values are optional following **+|-f**, **-F**, **-g**, **-i**, **+|-L**, **-o**, **+|-r**, **-s**, **-S**, **-T**, **-x** and **-z**. when you have no values for them be careful that the following character isn't ambiguous. For example, **-Fn** might represent the **-F** and **-n** options, or it might represent the **n** field identifier character following the **-F** option. When ambiguity is possible, start a new option with a ‘-’ character – e.g., “**-F -n**”. If the next option is a file name, follow the possibly ambiguous option with “**--**” – e.g., “**-F -- name**”.

Either the ‘+’ or the ‘-’ prefix may be applied to a group of options. Options that don't take on separate meanings for each prefix – e.g., **-i** – may be grouped under either prefix. Thus, for example, “**+M -i**” may be stated as “**+Mi**” and the group means the same as the separate options. Be careful of prefix grouping when one or more options in the group does take on separate meanings under different prefixes – e.g., **+|-M**; “**-iM**” is not the same request as “**-i +M**”. When in doubt, use separate options with appropriate prefixes.

-? -h These two equivalent options select a usage (help) output list. *Lsof* displays a shortened form of this output when it detects an error in the options supplied to it, after it has displayed messages explaining each error. (Escape the ‘?’ character as your shell requires.)

-a causes list selection options to be ANDed, as described above.

-A A is available on systems configured for AFS whose AFS kernel code is implemented via dynamic modules. It allows the *lsof* user to specify A as an alternate name list file where the kernel addresses of the dynamic modules might be found. See the *lsof* FAQ (The **FAQ** section gives its location.) for more information about dynamic modules, their symbols, and how they affect *lsof*.

-b causes *lsof* to avoid kernel functions that might block – *lstat(2)*, *readlink(2)*, and *stat(2)*.

See the **BLOCKS AND TIMEOUTS** and **AVOIDING KERNEL BLOCKS** sections for information on using this option.

-c c selects the listing of files for processes executing the command that begins with the characters of *c*. Multiple commands may be specified, using multiple **-c** options. The *c* are joined in a single ORed set before participating in AND option selection.

If *c* begins with a ‘‘’, then the following characters specify a command name whose processes are to be ignored (excluded.)

If *c* begins and ends with a slash (‘/’), the characters between the slashes are interpreted as a regular expression. Shell meta-characters in the regular expression must be quoted to prevent their interpretation by the shell. The closing slash may be followed by these modifiers:

- b the regular expression is a basic one.
- i ignore the case of letters.
- x the regular expression is an extended one
(default).

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information on basic and extended regular expressions.

The simple command specification is tested first. If that test fails, the command regular expression is applied. If the simple command test succeeds, the command regular expression test isn't made. This may result in “no command found for regex.” messages when *lsof*'s **-V** option is

specified.

+c w defines the maximum number of initial characters of the name, supplied by the UNIX dialect, of the UNIX command associated with a process to be printed in the COMMAND column. (The *lsof* default is nine.)

Note that many UNIX dialects do not supply all command name characters to *lsof* in the files and structures from which *lsof* obtains command name. Often dialects limit the number of characters supplied in those sources. For example, Linux 2.4.27 and Solaris 9 both limit command name length to 16 characters.

If *w* is zero ('0'), all command characters supplied to *lsof* by the UNIX dialect will be printed.

If *w* is less than the length of the column title, "COMMAND", it will be raised to that length.

-C disables the reporting of any path name components from the kernel's name cache. See the **KERNEL NAME CACHE** section for more information.

+d s causes *lsof* to search for all open instances of directory *s* and the files and directories it contains at its top level. **+d** does NOT descend the directory tree, rooted at *s*. The **+D D** option may be used to request a full-descent directory tree search, rooted at directory *D*.

Processing of the **+d** option does not follow symbolic links within *s* unless the **-x** or **-x I** option is also specified. Nor does it search for open files on file system mount points on subdirectories of *s* unless the **-x** or **-x f** option is also specified.

Note: the authority of the user of this option limits it to searching for files that the user has permission to examine with the system *stat*(2) function.

-d s specifies a list of file descriptors (FDs) to exclude from or include in the output listing. The file descriptors are specified in the comma-separated set *s* – e.g., "cwd,1,3", "^6,^2". (There should be no spaces in the set.)

The list is an exclusion list if all entries of the set begin with '^'. It is an inclusion list if no entry begins with '^'. Mixed lists are not permitted.

A file descriptor number range may be in the set as long as neither member is empty, both members are numbers, and the ending member is larger than the starting one – e.g., "0-7" or "3-10". Ranges may be specified for exclusion if they have the '^' prefix – e.g., "^0-7" excludes all file descriptors 0 through 7.

Multiple file descriptor numbers are joined in a single ORed set before participating in AND option selection.

When there are exclusion and inclusion members in the set, *lsof* reports them as errors and exits with a non-zero return code.

See the description of File Descriptor (FD) output values in the **OUTPUT** section for more information on file descriptor names.

+D D causes *lsof* to search for all open instances of directory *D* and all the files and directories it contains to its complete depth.

Processing of the **+D** option does not follow symbolic links within *D* unless the **-x** or **-x I** option is also specified. Nor does it search for open files on file system mount points on subdirectories of *D* unless the **-x** or **-x f** option is also specified.

Note: the authority of the user of this option limits it to searching for files that the user has permission to examine with the system *stat*(2) function.

Further note: *lsof* may process this option slowly and require a large amount of dynamic memory to do it. This is because it must descend the entire directory tree, rooted at *D*, calling *stat*(2) for each file and directory, building a list of all the files it finds, and searching that list for a match with every open file. When directory *D* is large, these steps can take a long time, so use this option prudently.

-D D directs *lsof*'s use of the device cache file. The use of this option is sometimes restricted. See the **DEVICE CACHE FILE** section and the sections that follow it for more information on this option.

-D must be followed by a function letter; the function letter may optionally be followed by a path name. *Lsof* recognizes these function letters:

- ? – report device cache file paths
- b** – build the device cache file
- i** – ignore the device cache file
- r** – read the device cache file
- u** – read and update the device cache file

The **b**, **r**, and **u** functions, accompanied by a path name, are sometimes restricted. When these functions are restricted, they will not appear in the description of the **-D** option that accompanies **-h** or **-?** option output. See the **DEVICE CACHE FILE** section and the sections that follow it for more information on these functions and when they're restricted.

The **?** function reports the read-only and write paths that *lsof* can use for the device cache file, the names of any environment variables whose values *lsof* will examine when forming the device cache file path, and the format for the personal device cache file path. (Escape the "?" character as your shell requires.)

When available, the **b**, **r**, and **u** functions may be followed by the device cache file's path. The standard default is *.lsof_hostname* in the home directory of the real user ID that executes *lsof*, but this could have been changed when *lsof* was configured and compiled. (The output of the **-h** and **-?** options show the current default prefix – e.g., ".*lsof*".) The suffix, *hostname*, is the first component of the host's name returned by *gethostname*(2).

When available, the **b** function directs *lsof* to build a new device cache file at the default or specified path.

The **i** function directs *lsof* to ignore the default device cache file and obtain its information about devices via direct calls to the kernel.

The **r** function directs *lsof* to read the device cache at the default or specified path, but prevents it from creating a new device cache file when none exists or the existing one is improperly structured. The **r** function, when specified without a path name, prevents *lsof* from updating an incorrect or outdated device cache file, or creating a new one in its place. The **r** function is always available when it is specified without a path name argument; it may be restricted by the permissions of the *lsof* process.

When available, the **u** function directs *lsof* to read the device cache file at the default or specified path, if possible, and to rebuild it, if necessary. This is the default device cache file function when no **-D** option has been specified.

+|-e s exempts the file system whose path name is *s* from being subjected to kernel function calls that might block. The **+e** option exempts *stat*(2), *lstat*(2) and most *readlink*(2) kernel function calls. The **-e** option exempts only *stat*(2) and *lstat*(2) kernel function calls. Multiple file systems may be specified with separate **+|-e** specifications and each may have *readlink*(2) calls exempted or not.

This option is currently implemented only for Linux.

CAUTION: this option can easily be mis-applied to other than the file system of interest, because it uses path name rather than the more reliable device and inode numbers. (Device and inode numbers are acquired via the potentially blocking *stat*(2) kernel call and are thus not available, but see the **+|-m m** option as a possible alternative way to supply device numbers.) **Use this option with great care and fully specify the path name of the file system to be exempted.**

When open files on exempted file systems are reported, it may not be possible to obtain all their information. Therefore, some information columns will be blank, the characters “UNKN” preface the values in the TYPE column, and the applicable exemption option is added in parentheses to the end of the NAME column. (Some device number information might be made available via the +|-m m option.)

- +|-E +E specifies that Linux pipe, Linux UNIX socket and Linux pseudoterminal files should be displayed with endpoint information and the files of the endpoints should also be displayed. Note: UNIX socket file endpoint information is only available when the compile flags line of -v output contains HASUXSOCKET, and pseudoterminal endpoint information is only available when the compile flags line contains HASPTYEPT.

Pipe endpoint information is displayed in the NAME column in the form “*PID,cmd,FDmode*”, where *PID* is the endpoint process ID; *cmd* is the endpoint process command; *FD* is the endpoint file’s descriptor; and *mode* is the endpoint file’s access mode.

Pseudoterminal endpoint information is displayed in the NAME column as “->/dev/ptsmin *PID,cmd,FDmode*” or “*PID,cmd,FDmode*”. The first form is for a master device; the second, for a slave device. *min* is a slave device’s minor device number; and *PID*, *cmd*, *FD* and *mode* are the same as with pipe endpoint information. Note: pseudoterminal endpoint information is only available when the compile flags line of -V output contains HASPTYEPT.

UNIX socket file endpoint information is displayed in the NAME column in the form “type=TYPE ->INO=INODE *PID,cmd,FDmode*”, where *TYPE* is the socket type; *INO* is the i-node number of the connected socket; and *PID*, *cmd*, *FD* and *mode* are the same as with pipe endpoint information. Note: UNIX socket file endpoint information is available only when the compile flags line of -v output contains HASUXSOCKET.

Multiple occurrences of this information can appear in a file’s NAME column.

-E specifies that Linux pipe and Linux UNIX socket files should be displayed with endpoint information, but not the files of the endpoints.

+|-f [cfgGn]

f by itself clarifies how path name arguments are to be interpreted. When followed by **c**, **f**, **g**, **G**, or **n** in any combination it specifies that the listing of kernel file structure information is to be enabled (+) or inhibited (-).

Normally a path name argument is taken to be a file system name if it matches a mounted-on directory name reported by *mount*(8), or if it represents a block device, named in the *mount* output and associated with a mounted directory name. When +f is specified, all path name arguments will be taken to be file system names, and *lsof* will complain if any are not. This can be useful, for example, when the file system name (mounted-on device) isn’t a block device. This happens for some CD-ROM file systems.

When -f is specified by itself, all path name arguments will be taken to be simple files. Thus, for example, the “-f -- /” arguments direct *lsof* to search for open files with a ‘/’ path name, not all open files in the ‘/’ (root) file system.

Be careful to make sure +f and -f are properly terminated and aren’t followed by a character (e.g., of the file or file system name) that might be taken as a parameter. For example, use “--” after +f and -f as in these examples.

```
$ lsof +f -- /file/system/name
$ lsof -f -- /file/name
```

The listing of information from kernel file structures, requested with the +f [cfgGn] option form, is normally inhibited, and is not available in whole or part for some dialects – e.g., /proc-based Linux kernels below 2.6.22. When the prefix to f is a plus sign (+), these characters request file structure information:

c	file structure use count (not Linux)
f	file structure address (not Linux)
g	file flag abbreviations (Linux 2.6.22 and up)
G	file flags in hexadecimal (Linux 2.6.22 and up)
n	file structure node address (not Linux)

When the prefix is minus ('-') the same characters disable the listing of the indicated values.

File structure addresses, use counts, flags, and node addresses may be used to detect more readily identical files inherited by child processes and identical files in use by different processes. *Lsof* column output can be sorted by output columns holding the values and listed to identify identical file use, or *lsof* field output can be parsed by an AWK or Perl post-filter script, or by a C program.

-F f specifies a character list, *f*, that selects the fields to be output for processing by another program, and the character that terminates each output field. Each field to be output is specified with a single character in *f*. The field terminator defaults to NL, but may be changed to NUL (000). See the **OUTPUT FOR OTHER PROGRAMS** section for a description of the field identification characters and the field output process.

When the field selection character list is empty, all standard fields are selected (except the raw device field, security context and zone field for compatibility reasons) and the NL field terminator is used.

When the field selection character list contains only a zero ('0'), all fields are selected (except the raw device field for compatibility reasons) and the NUL terminator character is used.

Other combinations of fields and their associated field terminator character must be set with explicit entries in *f*, as described in the **OUTPUT FOR OTHER PROGRAMS** section.

When a field selection character identifies an item *lsof* does not normally list – e.g., PPID, selected with **-R** – specification of the field character – e.g., “**-FR**” – also selects the listing of the item.

When the field selection character list contains the single character ‘?’ , *lsof* will display a help list of the field identification characters. (Escape the ‘?’ character as your shell requires.)

-g [s] excludes or selects the listing of files for the processes whose optional process group IDentification (PGID) numbers are in the comma-separated set *s* – e.g., “123” or “123,^456”. (There should be no spaces in the set.)

PGID numbers that begin with “^” (negation) represent exclusions.

Multiple PGID numbers are joined in a single ORed set before participating in AND option selection. However, PGID exclusions are applied without ORing or ANDing and take effect before other selection criteria are applied.

The **-g** option also enables the output display of PGID numbers. When specified without a PGID set that's all it does.

-i [i] selects the listing of files any of whose Internet address matches the address specified in *i*. If no address is specified, this option selects the listing of all Internet and x.25 (HP-UX) network files.

If **-i4** or **-i6** is specified with no following address, only files of the indicated IP version, IPv4 or IPv6, are displayed. (An IPv6 specification may be used only if the dialects supports IPv6, as indicated by “[46]” and “IPv[46]” in *lsof*'s **-h** or **-?** output.) Sequentially specifying **-i4**, followed by **-i6** is the same as specifying **-i**, and vice-versa. Specifying **-i4**, or **-i6** after **-i** is the same as specifying **-i4** or **-i6** by itself.

Multiple addresses (up to a limit of 100) may be specified with multiple **-i** options. (A port number or service name range is counted as one address.) They are joined in a single ORed set before participating in AND option selection.

An Internet address is specified in the form (Items in square brackets are optional.):

[46][*protocol*][@*hostname* | *hostaddr*][:*service* | *port*]

where:

46 specifies the IP version, IPv4 or IPv6
 that applies to the following address.
 '6' may be specified only if the UNIX
 dialect supports IPv6. If neither '4' nor
 '6' is specified, the following address
 applies to all IP versions.

protocol is a protocol name – **TCP, UDP**

hostname is an Internet host name. Unless a
 specific IP version is specified, open
 network files associated with host names
 of all versions will be selected.

hostaddr is a numeric Internet IPv4 address in
 dot form; or an IPv6 numeric address in
 colon form, enclosed in brackets, if the
 UNIX dialect supports IPv6. When an IP
 version is selected, only its numeric
 addresses may be specified.

service is an /etc/services name – e.g., **smtp** –
 or a list of them.

port is a port number, or a list of them.

IPv6 options may be used only if the UNIX dialect supports IPv6. To see if the dialect supports IPv6, run *lsof* and specify the **-h** or **-?** (help) option. If the displayed description of the **-i** option contains “[46]” and “IPv[46]”, IPv6 is supported.

IPv4 host names and addresses may not be specified if network file selection is limited to IPv6 with **-i 6**. IPv6 host names and addresses may not be specified if network file selection is limited to IPv4 with **-i 4**. When an open IPv4 network file’s address is mapped in an IPv6 address, the open file’s type will be IPv6, not IPv4, and its display will be selected by '6', not '4'.

At least one address component – **4, 6, protocol, hostname, hostaddr, or service** – must be supplied. The ‘@’ character, leading the host specification, is always required; as is the ‘:’, leading the port specification. Specify either *hostname* or *hostaddr*. Specify either *service* name list or *port* number list. If a *service* name list is specified, the *protocol* may also need to be specified if the TCP, UDP and UDPLITE port numbers for the service name are different. Use any case – lower or upper – for *protocol*.

Service names and *port* numbers may be combined in a list whose entries are separated by commas and whose numeric range entries are separated by minus signs. There may be no embedded spaces, and all service names must belong to the specified *protocol*. Since service names may contain embedded minus signs, the starting entry of a range can’t be a service name; it can be a port number, however.

Here are some sample addresses:

```
-i6 - IPv6 only
TCP:25 - TCP and port 25
@1.2.3.4 - Internet IPv4 host address 1.2.3.4
@[3ffe:1ebc::1]:1234 - Internet IPv6 host address
  3ffe:1ebc::1, port 1234
UDP:who - UDP who service port
TCP@lsof.itap:513 - TCP, port 513 and host name lsof.itap
tcp@foo:1-10,smtp,99 - TCP, ports 1 through 10,
  service name smtp, port 99, host name foo
```

`tcp@bar:1-smtp` – TCP, ports 1 through *smtp*, host bar
`:time` – either TCP, UDP or UDPLITE time service port

-K k selects the listing of tasks (threads) of processes, on dialects where task (thread) reporting is supported. (If help output – i.e., the output of the **-h** or **-?** options – shows this option, then task (thread) reporting is supported by the dialect.)

If **-K** is followed by a value, *k*, it must be “*i*”. That causes *lsof* to ignore tasks, particularly in the default, list-everything case when no other options are specified.

When **-K** and **-a** are both specified on Linux, and the tasks of a main process are selected by other options, the main process will also be listed as though it were a task, but without a task ID. (See the description of the TID column in the **OUTPUT** section.)

Where the FreeBSD version supports threads, all threads will be listed with their IDs.

In general threads and tasks inherit the files of the caller, but may close some and open others, so *lsof* always reports all the open files of threads and tasks.

-k k specifies a kernel name list file, *k*, in place of `/vmunix`, `/mach`, etc. **-k** is not available under AIX on the IBM RISC/System 6000.

-l inhibits the conversion of user ID numbers to login names. It is also useful when login name lookup is working improperly or slowly.

+|-L [l] enables (+) or disables (-) the listing of file link counts, where they are available – e.g., they aren't available for sockets, or most FIFOs and pipes.

When **+L** is specified without a following number, all link counts will be listed. When **-L** is specified (the default), no link counts will be listed.

When **+L** is followed by a number, only files having a link count less than that number will be listed. (No number may follow **-L**.) A specification of the form “**+L1**” will select open files that have been unlinked. A specification of the form “**+aL1 <file_system>**” will select un-linked open files on the specified file system.

For other link count comparisons, use field output (**-F**) and a post-processing script or program.

+|-m m specifies an alternate kernel memory file or activates mount table supplement processing.

The option form **-m m** specifies a kernel memory file, *m*, in place of `/dev/kmem` or `/dev/mem` – e.g., a crash dump file.

The option form **+m** requests that a mount supplement file be written to the standard output file. All other options are silently ignored.

There will be a line in the mount supplement file for each mounted file system, containing the mounted file system directory, followed by a single space, followed by the device number in hexadecimal "0x" format – e.g.,

/ 0x801

Lsof can use the mount supplement file to get device numbers for file systems when it can't get them via `stat(2)` or `lstat(2)`.

The option form **+m m** identifies *m* as a mount supplement file.

Note: the **+m** and **+m m** options are not available for all supported dialects. Check the output of *lsof*'s **-h** or **-?** options to see if the **+m** and **+m m** options are available.

+|-M Enables (+) or disables (-) the reporting of portmapper registrations for local TCP, UDP and UDPLITE ports, where port mapping is supported. (See the last paragraph of this option description for information about where portmapper registration reporting is supported.)

The default reporting mode is set by the *lsof* builder with the HASPMAPENABLED #define in the dialect's machine.h header file; *lsof* is distributed with the HASPMAPENABLED #define deactivated, so portmapper reporting is disabled by default and must be requested with **+M**.

Specifying *lsof*'s **-h** or **-?** option will report the default mode. Disabling portmapper registration when it is already disabled or enabling it when already enabled is acceptable. When portmapper registration reporting is enabled, *lsof* displays the portmapper registration (if any) for local TCP, UDP or UDPLITE ports in square brackets immediately following the port numbers or service names – e.g., “:1234[name]” or “:name[100083]”. The registration information may be a name or number, depending on what the registering program supplied to the portmapper when it registered the port.

When portmapper registration reporting is enabled, *lsof* may run a little more slowly or even become blocked when access to the portmapper becomes congested or stopped. Reverse the reporting mode to determine if portmapper registration reporting is slowing or blocking *lsof*.

For purposes of portmapper registration reporting *lsof* considers a TCP, UDP or UDPLITE port local if: it is found in the local part of its containing kernel structure; or if it is located in the foreign part of its containing kernel structure and the local and foreign Internet addresses are the same; or if it is located in the foreign part of its containing kernel structure and the foreign Internet address is INADDR_LOOPBACK (127.0.0.1). This rule may make *lsof* ignore some foreign ports on machines with multiple interfaces when the foreign Internet address is on a different interface from the local one.

See the *lsof* FAQ (The **FAQ** section gives its location.) for further discussion of portmapper registration reporting issues.

Portmapper registration reporting is supported only on dialects that have RPC header files. (Some Linux distributions with GlibC 2.14 do not have them.) When portmapper registration reporting is supported, the **-h** or **-?** help output will show the **+|-M** option.

- n** inhibits the conversion of network numbers to host names for network files. Inhibiting conversion may make *lsof* run faster. It is also useful when host name lookup is not working properly.
 - N** selects the listing of NFS files.
 - o** directs *lsof* to display file offset at all times. It causes the SIZE/OFF output column title to be changed to OFFSET. Note: on some UNIX dialects *lsof* can't obtain accurate or consistent file offset information from its kernel data sources, sometimes just for particular kinds of files (e.g., socket files.) Consult the *lsof* FAQ (The **FAQ** section gives its location.) for more information.
- The **-o** and **-s** options are mutually exclusive; they can't both be specified. When neither is specified, *lsof* displays whatever value – size or offset – is appropriate and available for the type of the file.
- o o** defines the number of decimal digits (*o*) to be printed after the “0t” for a file offset before the form is switched to “0x...”. A value of zero (unlimited) directs *lsof* to use the “0t” form for all offset output.

This option does NOT direct *lsof* to display offset at all times; specify **-o** (without a trailing number) to do that. **-o o** only specifies the number of digits after ‘‘0t’’ in either mixed size and offset or offset-only output. Thus, for example, to direct *lsof* to display offset at all times with a decimal digit count of 10, use:

```
-o -o 10
or
-oo10
```

The default number of digits allowed after “0t” is normally 8, but may have been changed by the *lsof* builder. Consult the description of the **-o o** option in the output of the **-h** or **-?** option to determine the default that is in effect.

- O** directs *lsof* to bypass the strategy it uses to avoid being blocked by some kernel operations – i.e., doing them in forked child processes. See the **BLOCKS AND TIMEOUTS** and **AVOIDING KERNEL BLOCKS** sections for more information on kernel operations that may block *lsof*.

While use of this option will reduce *lsof* startup overhead, it may also cause *lsof* to hang when the kernel doesn't respond to a function. Use this option cautiously.

-p s excludes or selects the listing of files for the processes whose optional process IDentification (PID) numbers are in the comma-separated set *s* – e.g., “123” or “123,^456”. (There should be no spaces in the set.)

PID numbers that begin with ‘^’ (negation) represent exclusions.

Multiple process ID numbers are joined in a single ORed set before participating in AND option selection. However, PID exclusions are applied without ORing or ANDing and take effect before other selection criteria are applied.

-P inhibits the conversion of port numbers to port names for network files. Inhibiting the conversion may make *lsof* run a little faster. It is also useful when port name lookup is not working properly.

+|-r [t[m<fmt>]]

puts *lsof* in repeat mode. There *lsof* lists open files as selected by other options, delays *t* seconds (default fifteen), then repeats the listing, delaying and listing repetitively until stopped by a condition defined by the prefix to the option.

If the prefix is a ‘-’, repeat mode is endless. *Lsof* must be terminated with an interrupt or quit signal.

If the prefix is ‘+’, repeat mode will end the first cycle no open files are listed – and of course when *lsof* is stopped with an interrupt or quit signal. When repeat mode ends because no files are listed, the process exit code will be zero if any open files were ever listed; one, if none were ever listed.

Lsof marks the end of each listing: if field output is in progress (the **-F** option has been specified), the default marker is ‘m’; otherwise the default marker is “=====”. The marker is followed by a NL character.

The optional “m<fmt>” argument specifies a format for the marker line. The <fmt> characters following ‘m’ are interpreted as a format specification to the *strftime(3)* function, when both it and the *localtime(3)* function are available in the dialect’s C library. Consult the *strftime(3)* documentation for what may appear in its format specification. Note that when field output is requested with the **-F** option, <fmt> cannot contain the NL format, “%n”. Note also that when <fmt> contains spaces or other characters that affect the shell’s interpretation of arguments, <fmt> must be quoted appropriately.

Repeat mode reduces *lsof* startup overhead, so it is more efficient to use this mode than to call *lsof* repetitively from a shell script, for example.

To use repeat mode most efficiently, accompany **+|-r** with specification of other *lsof* selection options, so the amount of kernel memory access *lsof* does will be kept to a minimum. Options that filter at the process level – e.g., **-c**, **-g**, **-p**, **-u** – are the most efficient selectors.

Repeat mode is useful when coupled with field output (see the **-F**, option description) and a supervising *awk* or *Perl* script, or a C program.

-R directs *lsof* to list the Parent Process IDentification number in the PPID column.

-s [p:s] *s* alone directs *lsof* to display file size at all times. It causes the SIZE/OFF output column title to be changed to SIZE. If the file does not have a size, nothing is displayed.

The optional **-s p:s** form is available only for selected dialects, and only when the **-h** or **-?** help output lists it.

When the optional form is available, the *s* may be followed by a protocol name (*p*), either TCP or UDP, a colon (‘:’) and a comma-separated protocol state name list, the option causes open TCP and UDP files to be excluded if their state name(s) are in the list (*s*) preceded by a ‘^’; or included if their name(s) are not preceded by a ‘^’.

Dialects that support this option may support only one protocol. When an unsupported protocol is specified, a message will be displayed indicating state names for the protocol are unavailable.

When an inclusion list is defined, only network files with state names in the list will be present in the *lsof* output. Thus, specifying one state name means that only network files with that lone state name will be listed.

Case is unimportant in the protocol or state names, but there may be no spaces and the colon (‘:’) separating the protocol name (*p*) and the state name list (*s*) is required.

If only TCP and UDP files are to be listed, as controlled by the specified exclusions and inclusions, the **-i** option must be specified, too. If only a single protocol's files are to be listed, add its name as an argument to the **-i** option.

For example, to list only network files with TCP state LISTEN, use:

```
-iTCP -sTCP:LISTEN
```

Or, for example, to list network files with all UDP states except Idle, use:

```
-iUDP -sUDP:^Idle
```

State names vary with UNIX dialects, so it's not possible to provide a complete list. Some common TCP state names are: CLOSED, IDLE, BOUND, LISTEN, ESTABLISHED, SYN_SENT, SYN_RCDV, ESTABLISHED, CLOSE_WAIT, FIN_WAIT1, CLOSING, LAST_ACK, FIN_WAIT_2, and TIME_WAIT. Two common UDP state names are Unbound and Idle.

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information on how to use protocol state exclusion and inclusion, including examples.

The **-o** (without a following decimal digit count) and **-s** option (without a following protocol and state name list) are mutually exclusive; they can't both be specified. When neither is specified, *lsof* displays whatever value – size or offset – is appropriate and available for the type of file.

Since some types of files don't have true sizes – sockets, FIFOs, pipes, etc. – *lsof* displays for their sizes the content amounts in their associated kernel buffers, if possible.

-S [t] specifies an optional time-out seconds value for kernel functions – *lstat(2)*, *readlink(2)*, and *stat(2)* – that might otherwise deadlock. The minimum for *t* is two; the default, fifteen; when no value is specified, the default is used.

See the **BLOCKS AND TIMEOUTS** section for more information.

-T [t] controls the reporting of some TCP/TPI information, also reported by *netstat(1)*, following the network addresses. In normal output the information appears in parentheses, each item except TCP or TPI state name identified by a keyword, followed by ‘=’, separated from others by a single space:

```
<TCP or TPI state name>
QR=<read queue length>
QS=<send queue length>
SO=<socket options and values>
SS=<socket states>
TF=<TCP flags and values>
WR=<window read length>
WW=<window write length>
```

Not all values are reported for all UNIX dialects. Items values (when available) are reported after the item name and ‘=’.

When the field output mode is in effect (See **OUTPUT FOR OTHER PROGRAMS.**) each item appears as a field with a ‘T’ leading character.

-T with no following key characters disables TCP/TPI information reporting.

-T with following characters selects the reporting of specific TCP/TPI information:

- f** selects reporting of socket options, states and values, and TCP flags and values.
- q** selects queue length reporting.
- s** selects connection state reporting.
- w** selects window size reporting.

Not all selections are enabled for some UNIX dialects. State may be selected for all dialects and is reported by default. The **-h** or **-?** help output for the **-T** option will show what selections may be used with the UNIX dialect.

When **-T** is used to select information – i.e., it is followed by one or more selection characters – the displaying of state is disabled by default, and it must be explicitly selected again in the characters following **-T**. (In effect, then, the default is equivalent to **-Ts**.) For example, if queue lengths and state are desired, use **-Tqs**.

Socket options, socket states, some socket values, TCP flags and one TCP value may be reported (when available in the UNIX dialect) in the form of the names that commonly appear after `SO_`, `so_`, `SS_`, `TCP_` and `TF_` in the dialect's header files – most often `<sys/socket.h>`, `<sys/socketvar.h>` and `<netinet/tcp_var.h>`. Consult those header files for the meaning of the flags, options, states and values.

“`SO=`” precedes socket options and values; “`SS=`”, socket states; and “`TF=`”, TCP flags and values.

If a flag or option has a value, the value will follow an ‘=’ and the name -- e.g., “`SO=LINGER=5`”, “`SO=QLIM=5`”, “`TF=MSS=512`”. The following seven values may be reported:

Name Reported	Description (Common Symbol)
KEEPALIVE	keep alive time (SO_KEEPALIVE)
LINGER	linger time (SO_LINGER)
MSS	maximum segment size (TCP_MAXSEG)
PQLEN	partial listen queue connections
QLEN	established listen queue connections
QLIM	established listen queue limit
RCVBUF	receive buffer length (SO_RCVBUF)
SNDBUF	send buffer length (SO_SNDBUF)

Details on what socket options and values, socket states, and TCP flags and values may be displayed for particular UNIX dialects may be found in the answer to the “Why doesn't lsof report socket options, socket states, and TCP flags and values for my dialect?” and “Why doesn't lsof report the partial listen queue connection count for my dialect?” questions in the *lsof FAQ* (The **FAQ** section gives its location.)

-t specifies that *lsof* should produce terse output with process identifiers only and no header – e.g., so that the output may be piped to *kill(1)*. **-t** selects the **-w** option.

-u s selects the listing of files for the user whose login names or user ID numbers are in the comma-separated set *s* – e.g., “abe”, or “548,root”. (There should be no spaces in the set.)

Multiple login names or user ID numbers are joined in a single ORed set before participating in AND option selection.

If a login name or user ID is preceded by a ‘^’, it becomes a negation – i.e., files of processes owned by the login name or user ID will never be listed. A negated login name or user ID

selection is neither ANDed nor ORed with other selections; it is applied before all other selections and absolutely excludes the listing of the files of the process. For example, to direct *lsof* to exclude the listing of files belonging to root processes, specify “*–u^root*” or “*–u^0*”.

- U*** selects the listing of UNIX domain socket files.
- v*** selects the listing of *lsof* version information, including: revision number; when the *lsof* binary was constructed; who constructed the binary and where; the name of the compiler used to construct the *lsof* binary; the version number of the compiler when readily available; the compiler and loader flags used to construct the *lsof* binary; and system information, typically the output of *uname*'s ***–a*** option.
- V*** directs *lsof* to indicate the items it was asked to list and failed to find – command names, file names, Internet addresses or files, login names, NFS files, PIDs, PGIDs, and UIDs.

When other options are ANDed to search options, or compile-time options restrict the listing of some files, *lsof* may not report that it failed to find a search item when an ANDed option or compile-time option prevents the listing of the open file containing the located search item.

For example, “*lsof –V –iTCP@foobar –a –d 999*” may not report a failure to locate open files at “TCP@foobar” and may not list any, if none have a file descriptor number of 999. A similar situation arises when HASSECURITY and HASNOSOCKSECURITY are defined at compile time and they prevent the listing of open files.

- +|–w*** Enables (+) or disables (–) the suppression of warning messages.
The ***lsof*** builder may choose to have warning messages disabled or enabled by default. The default warning message state is indicated in the output of the ***–h*** or ***–?*** option. Disabling warning messages when they are already disabled or enabling them when already enabled is acceptable.
The ***–t*** option selects the ***–w*** option.
- x [fl]*** may accompany the ***+d*** and ***+D*** options to direct their processing to cross over symbolic links and/or file system mount points encountered when scanning the directory (***+d***) or directory tree (***+D***).
If ***–x*** is specified by itself without a following parameter, cross-over processing of both symbolic links and file system mount points is enabled. Note that when ***–x*** is specified without a parameter, the next argument must begin with ‘–’ or ‘+’.
The optional ‘f’ parameter enables file system mount point cross-over processing; ‘l’, symbolic link cross-over processing.
The ***–x*** option may not be supplied without also supplying a ***+d*** or ***+D*** option.
- X*** This is a dialect-specific option.

AIX:

This IBM AIX RISC/System 6000 option requests the reporting of executed text file and shared library references.

WARNING: because this option uses the kernel readx() function, its use on a busy AIX system might cause an application process to hang so completely that it can neither be killed nor stopped. I have never seen this happen or had a report of its happening, but I think there is a remote possibility it could happen.

By default use of readx() is disabled. On AIX 5L and above *lsof* may need setuid-root permission to perform the actions this option requests.

The *lsof* builder may specify that the ***–X*** option be restricted to processes whose real UID is root. If that has been done, the ***–X*** option will not appear in the ***–h*** or ***–?*** help output unless the real UID of the *lsof* process is root. The default *lsof* distribution allows any UID to specify ***–X***, so by default it will appear in the help output.

When AIX readx() use is disabled, *lsof* may not be able to report information for all text and loader file references, but it may also avoid exacerbating an AIX kernel directory search kernel error, known as the Stale Segment ID bug.

The readx() function, used by *lsof* or any other program to access some sections of kernel virtual memory, can trigger the Stale Segment ID bug. It can cause the kernel's dir_search() function to believe erroneously that part of an in-memory copy of a file system directory has been zeroed. Another application process, distinct from *lsof*, asking the kernel to search the directory – e.g., by using *open(2)* – can cause dir_search() to loop forever, thus hanging the application process.

Consult the *lsof* FAQ (The **FAQ** section gives its location.) and the *00README* file of the *lsof* distribution for a more complete description of the Stale Segment ID bug, its APAR, and methods for defining readx() use when compiling *lsof*.

Linux:

This Linux option requests that *lsof* skip the reporting of information on all open TCP, UDP and UDPLITE IPv4 and IPv6 files.

This Linux option is most useful when the system has an extremely large number of open TCP, UDP and UDPLITE files, the processing of whose information in the */proc/net/tcp** and */proc/net/udp** files would take *lsof* a long time, and whose reporting is not of interest.

Use this option with care and only when you are sure that the information you want *lsof* to display isn't associated with open TCP, UDP or UDPLITE socket files.

Solaris 10 and above:

This Solaris 10 and above option requests the reporting of cached paths for files that have been deleted – i.e., removed with *rm(1)* or *unlink(2)*.

The cached path is followed by the string “(deleted)” to indicate that the path by which the file was opened has been deleted.

Because intervening changes made to the path – i.e., renames with *mv(1)* or *rename(2)* – are not recorded in the cached path, what *lsof* reports is only the path by which the file was opened, not its possibly different final path.

-z [z] specifies how Solaris 10 and higher zone information is to be handled.

Without a following argument – e.g., NO *z* – the option specifies that zone names are to be listed in the ZONE output column.

The **-z** option may be followed by a zone name, **z**. That causes *lsof* to list only open files for processes in that zone. Multiple **-z z** option and argument pairs may be specified to form a list of named zones. Any open file of any process in any of the zones will be listed, subject to other conditions specified by other options and arguments.

-Z [Z] specifies how SELinux security contexts are to be handled. It and 'Z' field output character support are inhibited when SELinux is disabled in the running Linux kernel. See **OUTPUT FOR OTHER PROGRAMS** for more information on the 'Z' field output character.

Without a following argument – e.g., NO *Z* – the option specifies that security contexts are to be listed in the SECURITY-CONTEXT output column.

The **-Z** option may be followed by a wildcard security context name, **Z**. That causes *lsof* to list only open files for processes in that security context. Multiple **-Z Z** option and argument pairs may be specified to form a list of security contexts. Any open file of any process in any of the security contexts will be listed, subject to other conditions specified by other options and arguments. Note that *Z* can be *A:B:C* or **:B:C* or *A:B:** or **:*:C* to match against the *A:B:C* context.

-- The double minus sign option is a marker that signals the end of the keyed options. It may be used, for example, when the first file name begins with a minus sign. It may also be used when

the absence of a value for the last keyed option must be signified by the presence of a minus sign in the following option and before the start of the file names.

names These are path names of specific files to list. Symbolic links are resolved before use. The first name may be separated from the preceding options with the “**--**” option.

If a *name* is the mounted-on directory of a file system or the device of the file system, *lsof* will list all the files open on the file system. To be considered a file system, the *name* must match a mounted-on directory name in *mount(8)* output, or match the name of a block device associated with a mounted-on directory name. The **+|-f** option may be used to force *lsof* to consider a *name* a file system identifier (**+f**) or a simple file (**-f**).

If *name* is a path to a directory that is not the mounted-on directory name of a file system, it is treated just as a regular file is treated – i.e., its listing is restricted to processes that have it open as a file or as a process-specific directory, such as the root or current working directory. To request that *lsof* look for open files inside a directory name, use the **+d s** and **+D D** options.

If a *name* is the base name of a family of multiplexed files – e.g., AIX’s */dev/pt[cs]* – *lsof* will list all the associated multiplexed files on the device that are open – e.g., */dev/pt[cs]/1*, */dev/pt[cs]/2*, etc.

If a *name* is a UNIX domain socket name, *lsof* will usually search for it by the characters of the name alone – exactly as it is specified and is recorded in the kernel socket structure. (See the next paragraph for an exception to that rule for Linux.) Specifying a relative path – e.g., *./file* – in place of the file’s absolute path – e.g., */tmp/file* – won’t work because *lsof* must match the characters you specify with what it finds in the kernel UNIX domain socket structures.

If a *name* is a Linux UNIX domain socket name, in one case *lsof* is able to search for it by its device and inode number, allowing *name* to be a relative path. The case requires that the absolute path -- i.e., one beginning with a slash (‘/’) be used by the process that created the socket, and hence be stored in the */proc/net/unix* file; and it requires that *lsof* be able to obtain the device and node numbers of both the absolute path in */proc/net/unix* and *name* via successful *stat(2)* system calls. When those conditions are met, *lsof* will be able to search for the UNIX domain socket when some path to it is specified in *name*. Thus, for example, if the path is */dev/log*, and an *lsof* search is initiated when the working directory is */dev*, then *name* could be *.log*.

If a *name* is none of the above, *lsof* will list any open files whose device and inode match that of the specified path *name*.

If you have also specified the **-b** option, the only *names* you may safely specify are file systems for which your mount table supplies alternate device numbers. See the **AVOIDING KERNEL BLOCKS** and **ALTERNATE DEVICE NUMBERS** sections for more information.

Multiple file names are joined in a single ORed set before participating in AND option selection.

AFS

lsof supports the recognition of AFS files for these dialects (and AFS versions):

- AIX 4.1.4 (AFS 3.4a)
- HP-UX 9.0.5 (AFS 3.4a)
- Linux 1.2.13 (AFS 3.3)
- Solaris 2.[56] (AFS 3.4a)

It may recognize AFS files on other versions of these dialects, but has not been tested there. Depending on how AFS is implemented, *lsof* may recognize AFS files in other dialects, or may have difficulties recognizing AFS files in the supported dialects.

lsof may have trouble identifying all aspects of AFS files in supported dialects when AFS kernel support is implemented via dynamic modules whose addresses do not appear in the kernel’s variable name list. In that case, *lsof* may have to guess at the identity of AFS files, and might not be able to obtain volume

information from the kernel that is needed for calculating AFS volume node numbers. When *lsof* can't compute volume node numbers, it reports blank in the NODE column.

The **-A** option is available in some dialect implementations of *lsof* for specifying the name list file where dynamic module kernel addresses may be found. When this option is available, it will be listed in the *lsof* help output, presented in response to the **-h** or **-?**

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information about dynamic modules, their symbols, and how they affect *lsof* options.

Because AFS path lookups don't seem to participate in the kernel's name cache operations, *lsof* can't identify path name components for AFS files.

SECURITY

Lsof has three features that may cause security concerns. First, its default compilation mode allows anyone to list all open files with it. Second, by default it creates a user-readable and user-writable device cache file in the home directory of the real user ID that executes *lsof*. (The list-all-open-files and device cache features may be disabled when *lsof* is compiled.) Third, its **-k** and **-m** options name alternate kernel name list or memory files.

Restricting the listing of all open files is controlled by the compile-time HASSECURITY and HASNOSOCKSECURITY options. When HASSECURITY is defined, *lsof* will allow only the root user to list all open files. The non-root user may list only open files of processes with the same user IDentification number as the real user ID number of the *lsof* process (the one that its user logged on with).

However, if HASSECURITY and HASNOSOCKSECURITY are both defined, anyone may list open socket files, provided they are selected with the **-i** option.

When HASSECURITY is not defined, anyone may list all open files.

Help output, presented in response to the **-h** or **-?** option, gives the status of the HASSECURITY and HASNOSOCKSECURITY definitions.

See the **Security** section of the *00README* file of the *lsof* distribution for information on building *lsof* with the HASSECURITY and HASNOSOCKSECURITY options enabled.

Creation and use of a user-readable and user-writable device cache file is controlled by the compile-time HASDCACHE option. See the **DEVICE CACHE FILE** section and the sections that follow it for details on how its path is formed. For security considerations it is important to note that in the default *lsof* distribution, if the real user ID under which *lsof* is executed is root, the device cache file will be written in root's home directory – e.g., */* or */root*. When HASDCACHE is not defined, *lsof* does not write or attempt to read a device cache file.

When HASDCACHE is defined, the *lsof* help output, presented in response to the **-h**, **-D?**, or **-?** options, will provide device cache file handling information. When HASDCACHE is not defined, the **-h** or **-?** output will have no **-D** option description.

Before you decide to disable the device cache file feature – enabling it improves the performance of *lsof* by reducing the startup overhead of examining all the nodes in */dev* (or */devices*) – read the discussion of it in the *00DCACHE* file of the *lsof* distribution and the *lsof* FAQ (The **FAQ** section gives its location.)

WHEN IN DOUBT, YOU CAN TEMPORARILY DISABLE THE USE OF THE DEVICE CACHE FILE WITH THE **-Di OPTION.**

When *lsof* user declares alternate kernel name list or memory files with the **-k** and **-m** options, *lsof* checks the user's authority to read them with *access(2)*. This is intended to prevent whatever special power *lsof*'s modes might confer on it from letting it read files not normally accessible via the authority of the real user ID.

OUTPUT

This section describes the information *lsof* lists for each open file. See the **OUTPUT FOR OTHER PROGRAMS** section for additional information on output that can be processed by another program.

Lsof only outputs printable (declared so by *isprint(3)*) 8 bit characters. Non-printable characters are

printed in one of three forms: the C “\[bfrnt]” form; the control character “`” form (e.g., “^@”); or hexa-decimal leading “\x” form (e.g., “\xab”). Space is non-printable in the COMMAND column (“\x20”) and printable elsewhere.

For some dialects – if HASSETLOCALE is defined in the dialect’s machine.h header file – *lsof* will print the extended 8 bit characters of a language locale. The *lsof* process must be supplied a language locale environment variable (e.g., LANG) whose value represents a known language locale in which the extended characters are considered printable by *isprint(3)*. Otherwise *lsof* considers the extended characters non-printable and prints them according to its rules for non-printable characters, stated above. Consult your dialect’s *setlocale(3)* man page for the names of other environment variables that may be used in place of LANG – e.g., LC_ALL, LC_CTYPE, etc.

Lsof’s language locale support for a dialect also covers wide characters – e.g., UTF-8 – when HASSETLOCALE and HASWIDECHAR are defined in the dialect’s machine.h header file, and when a suitable language locale has been defined in the appropriate environment variable for the *lsof* process. Wide characters are printable under those conditions if *iswprint(3)* reports them to be. If HASSETLOCALE, HASWIDECHAR and a suitable language locale aren’t defined, or if *iswprint(3)* reports wide characters that aren’t printable, *lsof* considers the wide characters non-printable and prints each of their 8 bits according to its rules for non-printable characters, stated above.

Consult the answers to the "Language locale support" questions in the *lsof* FAQ (The **FAQ** section gives its location.) for more information.

Lsof dynamically sizes the output columns each time it runs, guaranteeing that each column is a minimum size. It also guarantees that each column is separated from its predecessor by at least one space.

COMMAND contains the first nine characters of the name of the UNIX command associated with the process. If a non-zero *w* value is specified to the **+c w** option, the column contains the first *w* characters of the name of the UNIX command associated with the process up to the limit of characters supplied to *lsof* by the UNIX dialect. (See the description of the **+c w** command or the *lsof* FAQ for more information. The **FAQ** section gives its location.)

If *w* is less than the length of the column title, “COMMAND”, it will be raised to that length.

If a zero *w* value is specified to the **+c w** option, the column contains all the characters of the name of the UNIX command associated with the process.

All command name characters maintained by the kernel in its structures are displayed in field output when the command name descriptor (‘c’) is specified. See the **OUTPUT FOR OTHER COMMANDS** section for information on selecting field output and the associated command name descriptor.

PID is the Process IDentification number of the process.

TID is the task (thread) IDentification number, if task (thread) reporting is supported by the dialect and a task (thread) is being listed. (If help output – i.e., the output of the **-h** or **-?** options – shows this option, then task (thread) reporting is supported by the dialect.)

A blank TID column in Linux indicates a process – i.e., a non-task.

TASKCMD is the task command name. Generally this will be the same as the process named in the COMMAND column, but some task implementations (e.g., Linux) permit a task to change its command name.

The TASKCMD column width is subject to the same size limitation as the COMMAND column.

ZONE is the Solaris 10 and higher zone name. This column must be selected with the **-z** option.

SECURITY-CONTEXT

is the SELinux security context. This column must be selected with the **-Z** option. Note that the **-Z** option is inhibited when SELinux is disabled in the running Linux kernel.

PPID

is the Parent Process IDentification number of the process. It is only displayed when the **-R** option has been specified.

PGID

is the process group IDentification number associated with the process. It is only displayed when the **-g** option has been specified.

USER

is the user ID number or login name of the user to whom the process belongs, usually the same as reported by *ps*(1). However, on Linux USER is the user ID number or login that owns the directory in /proc where *lsof* finds information about the process. Usually that is the same value reported by *ps*(1), but may differ when the process has changed its effective user ID. (See the **-I** option description for information on when a user ID number or login name is displayed.)

FD

is the File Descriptor number of the file or:

cwd	current working directory;
Lnn	library references (AIX);
err	FD information error (see NAME column);
jld	jail directory (FreeBSD);
ltx	shared library text (code and data);
Mxx	hex memory-mapped type number xx.
m86	DOS Merge mapped file;
mem	memory-mapped file;
mmap	memory-mapped device;
pd	parent directory;
rtd	root directory;
tr	kernel trace file (OpenBSD);
txt	program text (code and data);
v86	VP/ix mapped file;

FD is followed by one of these characters, describing the mode under which the file is open:

- r** for read access;
- w** for write access;
- u** for read and write access;
- space if mode unknown and no lock character follows;
- '-' if mode unknown and lock character follows.

The mode character is followed by one of these lock characters, describing the type of lock applied to the file:

- N** for a Solaris NFS lock of unknown type;
- r** for read lock on part of the file;
- R** for a read lock on the entire file;
- w** for a write lock on part of the file;
- W** for a write lock on the entire file;
- u** for a read and write lock of any length;
- U** for a lock of unknown type;
- x** for an SCO OpenServer Xenix lock on part of the file;
- X** for an SCO OpenServer Xenix lock on the entire file;
- space if there is no lock.

	See the LOCKS section for more information on the lock information character.
TYPE	The FD column contents constitutes a single field for parsing in post-processing scripts. is the type of the node associated with the file – e.g., GDIR, GREG, VDIR, VREG, etc. or “IPv4” for an IPv4 socket; or “IPv6” for an open IPv6 network file – even if its address is IPv4, mapped in an IPv6 address; or “ax25” for a Linux AX.25 socket; or “inet” for an Internet domain socket; or “lla” for a HP-UX link level access file; or “rte” for an AF_ROUTE socket; or “sock” for a socket of unknown domain; or “unix” for a UNIX domain socket; or “x.25” for an HP-UX x.25 socket; or “BLK” for a block special file; or “CHR” for a character special file; or “DEL” for a Linux map file that has been deleted; or “DIR” for a directory; or “DOOR” for a VDOOR file; or “FIFO” for a FIFO special file; or “KQUEUE” for a BSD style kernel event queue file; or “LINK” for a symbolic link file; or “MPB” for a multiplexed block file; or “MPC” for a multiplexed character file; or “NOFD” for a Linux /proc/<PID>/fd directory that can't be opened -- the directory path appears in the NAME column, followed by an error message; or “PAS” for a /proc/as file; or “PAXV” for a /proc/auxv file; or “PCRE” for a /proc/cred file; or “PCTL” for a /proc control file; or “PCUR” for the current /proc process; or “PCWD” for a /proc current working directory; or “PDIR” for a /proc directory; or “PETY” for a /proc executable type (<i>etype</i>); or “PFD” for a /proc file descriptor; or “PFDR” for a /proc file descriptor directory; or “PFIL” for an executable /proc file; or “PFPR” for a /proc FP register set; or “PGD” for a /proc/pagedata file; or “PGID” for a /proc group notifier file;

or “PIPE” for pipes;
or “PLC” for a */proc/lwpctl* file;
or “PLDR” for a */proc/lpw* directory;
or “PLDT” for a */proc/ldt* file;
or “PLPI” for a */proc/lpsinfo* file;
or “PLST” for a */proc/lstatus* file;
or “PLU” for a */proc/lusage* file;
or “PLWG” for a */proc/gwindows* file;
or “PLWI” for a */proc/lwpsinfo* file;
or “PLWS” for a */proc/lwpstatus* file;
or “PLWU” for a */proc/lwpusage* file;
or “PLWX” for a */proc/xregs* file;
or “PMAP” for a */proc* map file (*map*);
or “PMEM” for a */proc* memory image file;
or “PNTF” for a */proc* process notifier file;
or “POBJ” for a */proc/object* file;
or “PODR” for a */proc/object* directory;
or “POLP” for an old format */proc* light weight process file;
or “POPF” for an old format */proc* PID file;
or “POPG” for an old format */proc* page data file;
or “PORT” for a SYSV named pipe;
or “PREG” for a */proc* register file;
or “PRMP” for a */proc/rmap* file;
or “PRTD” for a */proc* root directory;
or “PSGA” for a */proc/sigact* file;
or “PSIN” for a */proc/psinfo* file;
or “PSTA” for a */proc* status file;
or “PSXSEM” for a POSIX semaphore file;
or “PSXSHM” for a POSIX shared memory file;
or “PTS” for a */dev/pts* file;
or “PUSG” for a */proc/usage* file;
or “PW” for a */proc/watch* file;
or “PXMP” for a */proc/xmap* file;
or “REG” for a regular file;
or “SMT” for a shared memory transport file;
or “STSO” for a stream socket;
or “UNNM” for an unnamed type file;
or “XNAM” for an OpenServer Xenix special file of unknown type;

	or “XSEM” for an OpenServer Xenix semaphore file;
	or “XSD” for an OpenServer Xenix shared data file;
	or the four type number octets if the corresponding name isn't known.
FILE-ADDR	contains the kernel file structure address when f has been specified to +f ;
FCT	contains the file reference count from the kernel file structure when c has been specified to +f ;
FILE-FLAG	when g or G has been specified to +f , this field contains the contents of the f_flag[s] member of the kernel file structure and the kernel's per-process open file flags (if available); ‘G’ causes them to be displayed in hexadecimal; ‘g’, as short-hand names; two lists may be displayed with entries separated by commas, the lists separated by a semi-colon (‘;’); the first list may contain short-hand names for f_flag[s] values from the following table:
AIO	asynchronous I/O (e.g., FAIO)
AP	append
ASYN	asynchronous I/O (e.g., FASYNC)
BAS	block, test, and set in use
BKIU	block if in use
BL	use block offsets
BSK	block seek
CA	copy avoid
CIO	concurrent I/O
CLON	clone
CLRD	CL read
CR	create
DF	defer
DFI	defer IND
DFLU	data flush
DIR	direct
DLY	delay
DOCL	do clone
DSYN	data-only integrity
DTY	must be a directory
EVO	event only
EX	open for exec
EXCL	exclusive open
FSYN	synchronous writes
GCDF	defer during unp_gc() (AIX)
GCMK	mark during unp_gc() (AIX)
GTTY	accessed via /dev/tty
HUP	HUP in progress
KERN	kernel
KIOC	kernel-issued ioctl
LCK	has lock
LG	large file
MBLK	stream message block
MK	mark
MNT	mount
MSYN	multiplex synchronization
NATM	don't update atime
NB	non-blocking I/O
NBDR	no BDRM check
NBIO	SYSV non-blocking I/O

NBF	n-buffering in effect
NC	no cache
ND	no delay
NDSY	no data synchronization
NET	network
NFLK	don't follow links
NMFS	NM file system
NOTO	disable background stop
NSH	no share
NTTY	no controlling TTY
OLRM	OLR mirror
PAIO	POSIX asynchronous I/O
PP	POSIX pipe
R	read
RC	file and record locking cache
REV	revoked
RSH	shared read
RSYN	read synchronization
RW	read and write access
SL	shared lock
SNAP	cooked snapshot
SOCK	socket
SQSH	Sequent shared set on open
SQSV	Sequent SVM set on open
SQR	Sequent set repair on open
SQS1	Sequent full shared open
SQS2	Sequent partial shared open
STPI	stop I/O
SWR	synchronous read
SYN	file integrity while writing
TCPM	avoid TCP collision
TR	truncate
W	write
WKUP	parallel I/O synchronization
WTG	parallel I/O synchronization
VH	vhangup pending
VTXT	virtual text
XL	exclusive lock

this list of names was derived from F* #define's in dialect header files <fcntl.h>, <linux</fs.h>, <sys/fcntl.c>, <sys/fcntlcom.h>, and <sys/file.h>; see the lsof.h header file for a list showing the correspondence between the above short-hand names and the header file definitions;

the second list (after the semicolon) may contain short-hand names for kernel per-process open file flags from this table:

ALLC	allocated
BR	the file has been read
BHUP	activity stopped by SIGHUP
BW	the file has been written
CLSG	closing
CX	close-on-exec (see fcntl(F_SETFD))
LCK	lock was applied
MP	memory-mapped
OPIP	open pending – in progress

	RSVW	reserved wait
	SHMT	UF_FSHMAT set (AIX)
	USE	in use (multi-threaded)
NODE-ID	(or INODE-ADDR for some dialects)	contains a unique identifier for the file node (usually the kernel vnode or inode address, but also occasionally a concatenation of device and node number) when n has been specified to +f;
DEVICE		contains the device numbers, separated by commas, for a character special, block special, regular, directory or NFS file; or "memory" for a memory file system node under Tru64 UNIX; or the address of the private data area of a Solaris socket stream; or a kernel reference address that identifies the file (The kernel reference address may be used for FIFO's, for example.); or the base address or device name of a Linux AX.25 socket device. Usually only the lower thirty two bits of Tru64 UNIX kernel addresses are displayed.
SIZE, SIZE/OFF, or OFFSET		is the size of the file or the file offset in bytes. A value is displayed in this column only if it is available. <i>Lsof</i> displays whatever value – size or offset – is appropriate for the type of the file and the version of <i>Lsof</i> . On some UNIX dialects <i>Lsof</i> can't obtain accurate or consistent file offset information from its kernel data sources, sometimes just for particular kinds of files (e.g., socket files.) In other cases, files don't have true sizes – e.g., sockets, FIFOs, pipes – so <i>Lsof</i> displays for their sizes the content amounts it finds in their kernel buffer descriptors (e.g., socket buffer size counts or TCP/IP window sizes.) Consult the <i>Lsof</i> FAQ (The FAQ section gives its location.) for more information. The file size is displayed in decimal; the offset is normally displayed in decimal with a leading "0t" if it contains 8 digits or less; in hexadecimal with a leading "0x" if it is longer than 8 digits. (Consult the -o o option description for information on when 8 might default to some other value.) Thus the leading "0t" and "0x" identify an offset when the column may contain both a size and an offset (i.e., its title is SIZE/OFF). If the -o option is specified, <i>Lsof</i> always displays the file offset (or nothing if no offset is available) and labels the column OFFSET. The offset always begins with "0t" or "0x" as described above. The <i>Lsof</i> user can control the switch from "0t" to "0x" with the -o o option. Consult its description for more information. If the -s option is specified, <i>Lsof</i> always displays the file size (or nothing if no size is available) and labels the column SIZE. The -o and -s options are mutually exclusive; they can't both be specified. For files that don't have a fixed size – e.g., don't reside on a disk device – <i>Lsof</i> will display appropriate information about the current size or position of the file if it is available in the kernel structures that define the file.
NLINK		contains the file link count when +L has been specified;
NODE		is the node number of a local file; or the inode number of an NFS file in the server host; or the Internet protocol type – e.g, "TCP";

	<p>or “STR” for a stream;</p> <p>or “CCITT” for an HP–UX x.25 socket;</p> <p>or the IRQ or inode number of a Linux AX.25 socket device.</p>
NAME	<p>is the name of the mount point and file system on which the file resides;</p> <p>or the name of a file specified in the <i>names</i> option (after any symbolic links have been resolved);</p> <p>or the name of a character special or block special device;</p> <p>or the local and remote Internet addresses of a network file; the local host name or IP number is followed by a colon (‘:’), the port, “->”, and the two-part remote address; IP addresses may be reported as numbers or names, depending on the + –M, –n, and –P options; colon-separated IPv6 numbers are enclosed in square brackets; IPv4 IN-ADDR_ANY and IPv6 IN6_IS_ADDR_UNSPECIFIED addresses, and zero port numbers are represented by an asterisk (*); a UDP destination address may be followed by the amount of time elapsed since the last packet was sent to the destination; TCP, UDP and UDPLITE remote addresses may be followed by TCP/TPI information in parentheses – state (e.g., “(ESTABLISHED)”, “(Unbound)”), queue sizes, and window sizes (not all dialects) – in a fashion similar to what <i>netstat</i>(1) reports; see the –T option description or the description of the TCP/TPI field in OUTPUT FOR OTHER PROGRAMS for more information on state, queue size, and window size;</p> <p>or the address or name of a UNIX domain socket, possibly including a stream clone device name, a file system object’s path name, local and foreign kernel addresses, socket pair information, and a bound vnode address;</p> <p>or the local and remote mount point names of an NFS file;</p> <p>or “STR”, followed by the stream name;</p> <p>or a stream character device name, followed by “->” and the stream name or a list of stream module names, separated by “->”;</p> <p>or “STR:” followed by the SCO OpenServer stream device and module names, separated by “->”;</p> <p>or system directory name, “—”, and as many components of the path name as <i>lsof</i> can find in the kernel’s name cache for selected dialects (See the KERNEL NAME CACHE section for more information.);</p> <p>or “PIPE->”, followed by a Solaris kernel pipe destination address;</p> <p>or “COMMON:”, followed by the vnode device information structure’s device name, for a Solaris common vnode;</p> <p>or the address family, followed by a slash (/), followed by fourteen comma-separated bytes of a non-Internet raw socket address;</p> <p>or the HP–UX x.25 local address, followed by the virtual connection number (if any), followed by the remote address (if any);</p> <p>or “(dead)” for disassociated Tru64 UNIX files – typically terminal files that have been flagged with the TIOCNNOTTY ioctl and closed by daemons;</p> <p>or “rd=<offset>” and “wr=<offset>” for the values of the read and write offsets of a FIFO;</p> <p>or “clone n:/dev/event” for SCO OpenServer file clones of the /dev/event device, where n is the minor device number of the file;</p> <p>or “(socketpair: n)” for a Solaris 2.6, 8, 9 or 10 UNIX domain socket, created by the <i>socketpair</i>(3N) network function;</p>

or “no PCB” for socket files that do not have a protocol block associated with them, optionally followed by “, CANTSENDMORE” if sending on the socket has been disabled, or “, CANTRCVMORE” if receiving on the socket has been disabled (e.g., by the *shutdown(2)* function);

or the local and remote addresses of a Linux IPX socket file in the form <net>:[<node>:]<port>, followed in parentheses by the transmit and receive queue sizes, and the connection state;

or “dgram” or “stream” for the type UnixWare 7.1.1 and above in-kernel UNIX domain sockets, followed by a colon (‘:’) and the local path name when available, followed by “->” and the remote path name or kernel socket address in hexadecimal when available;

or the association value, association index, endpoint value, local address, local port, remote address and remote port for Linux SCTP sockets;

or “protocol:” followed by the Linux socket’s protocol attribute.

For dialects that support a “namefs” file system, allowing one file to be attached to another with *fattach(3C)*, *lsof* will add “(FA:<address1><direction><address2>)” to the NAME column. <address1> and <address2> are hexadecimal vnode addresses. <direction> will be “<–” if <address2> has been fattach’ed to this vnode whose address is <address1>; and “->” if <address1>, the vnode address of this vnode, has been fattach’ed to <address2>. <address1> may be omitted if it already appears in the DEVICE column.

Lsof may add two parenthetical notes to the NAME column for open Solaris 10 files: “(?)” if *lsof* considers the path name of questionable accuracy; and “(deleted)” if the **-X** option has been specified and *lsof* detects the open file’s path name has been deleted. Consult the *lsof* FAQ (The **FAQ** section gives its location.) for more information on these NAME column additions.

LOCKS

Lsof can’t adequately report the wide variety of UNIX dialect file locks in a single character. What it reports in a single character is a compromise between the information it finds in the kernel and the limitations of the reporting format.

Moreover, when a process holds several byte level locks on a file, *lsof* only reports the status of the first lock it encounters. If it is a byte level lock, then the lock character will be reported in lower case – i.e., ‘r’, ‘w’, or ‘x’ – rather than the upper case equivalent reported for a full file lock.

Generally *lsof* can only report on locks held by local processes on local files. When a local process sets a lock on a remotely mounted (e.g., NFS) file, the remote server host usually records the lock state. One exception is Solaris – at some patch levels of 2.3, and in all versions above 2.4, the Solaris kernel records information on remote locks in local structures.

Lsof has trouble reporting locks for some UNIX dialects. Consult the **BUGS** section of this manual page or the *lsof* FAQ (The **FAQ** section gives its location.) for more information.

OUTPUT FOR OTHER PROGRAMS

When the **-F** option is specified, *lsof* produces output that is suitable for processing by another program – e.g, an *awk* or *Perl* script, or a C program.

Each unit of information is output in a field that is identified with a leading character and terminated by a NL (012) (or a NUL (000) if the 0 (zero) field identifier character is specified.) The data of the field follows immediately after the field identification character and extends to the field terminator.

It is possible to think of field output as process and file sets. A process set begins with a field whose identifier is ‘p’ (for process IDentifier (PID)). It extends to the beginning of the next PID field or the beginning of the first file set of the process, whichever comes first. Included in the process set are fields that identify the command, the process group IDentification (PGID) number, the task (thread) ID (TID), and the user ID (UID) number or login name.

A file set begins with a field whose identifier is ‘f’ (for file descriptor). It is followed by lines that describe the file’s access mode, lock state, type, device, size, offset, inode, protocol, name and stream module names. It extends to the beginning of the next file or process set, whichever comes first.

When the NUL (000) field terminator has been selected with the 0 (zero) field identifier character, *lsof* ends each process and file set with a NL (012) character.

Lsof always produces one field, the PID ('p') field. All other fields may be declared optionally in the field identifier character list that follows the **-F** option. When a field selection character identifies an item *lsof* does not normally list – e.g., PPID, selected with **-R** – specification of the field character – e.g., “**-FR**” – also selects the listing of the item.

It is entirely possible to select a set of fields that cannot easily be parsed – e.g., if the field descriptor field is not selected, it may be difficult to identify file sets. To help you avoid this difficulty, *lsof* supports the **-F0** option; it selects the output of all fields with NL terminators (the **-F0** option pair selects the output of all fields with NUL terminators). For compatibility reasons neither **-F** nor **-F0** select the raw device field.

These are the fields that *lsof* will produce. The single character listed first is the field identifier.

a	file access mode
c	process command name (all characters from proc or user structure)
C	file structure share count
d	file's device character code
D	file's major/minor device number (0x<hexadecimal>)
f	file descriptor (always selected)
F	file structure address (0x<hexadecimal>)
G	file flaGs (0x<hexadecimal>; names if +fg follows)
g	process group ID
i	file's inode number
K	tasK ID
k	link count
l	file's lock status
L	process login name
m	marker between repeated output
M	the task comMand name
n	file name, comment, Internet address
N	node identifier (0x<hexadecimal>)
o	file's offset (decimal)
p	process ID (always selected)
P	protocol name
r	raw device number (0x<hexadecimal>)
R	parent process ID
s	file's size (decimal)
S	file's stream identification
t	file's type
T	TCP/TPI information, identified by prefixes (the '=' is part of the prefix): QR=<read queue size> QS=<send queue size> SO=<socket options and values> (not all dialects) SS=<socket states> (not all dialects) ST=<connection state> TF=<TCP flags and values> (not all dialects) WR=<window read size> (not all dialects) WW=<window write size> (not all dialects) (TCP/TPI information isn't reported for all supported UNIX dialects. The -h or -? help output for the -T option will show what TCP/TPI reporting can be requested.)
u	process user ID

z	Solaris 10 and higher zone name
Z	SELinux security context (inhibited when SELinux is disabled)
0	use NUL field terminator character in place of NL
1–9	dialect-specific field identifiers (The output of -F? identifies the information to be found in dialect-specific fields.)

You can get on-line help information on these characters and their descriptions by specifying the **-F?** option pair. (Escape the “?” character as your shell requires.) Additional information on field content can be found in the **OUTPUT** section.

As an example, “**-F pcfn**” will select the process ID ('p'), command name ('c'), file descriptor ('f') and file name ('n') fields with an NL field terminator character; “**-F pcfn0**” selects the same output with a NUL (000) field terminator character.

Lsof doesn't produce all fields for every process or file set, only those that are available. Some fields are mutually exclusive: file device characters and file major/minor device numbers; file inode number and protocol name; file name and stream identification; file size and offset. One or the other member of these mutually exclusive sets will appear in field output, but not both.

Normally *lsof* ends each field with a NL (012) character. The 0 (zero) field identifier character may be specified to change the field terminator character to a NUL (000). A NUL terminator may be easier to process with *xargs (1)*, for example, or with programs whose quoting mechanisms may not easily cope with the range of characters in the field output. When the NUL field terminator is in use, *lsof* ends each process and file set with a NL (012).

Three aids to producing programs that can process *lsof* field output are included in the *lsof* distribution. The first is a C header file, *lsof_fields.h*, that contains symbols for the field identification characters, indexes for storing them in a table, and explanation strings that may be compiled into programs. *Lsof* uses this header file.

The second aid is a set of sample scripts that process field output, written in *awk*, *Perl* 4, and *Perl* 5. They're located in the *scripts* subdirectory of the *lsof* distribution.

The third aid is the C library used for the *lsof* test suite. The test suite is written in C and uses field output to validate the correct operation of *lsof*. The library can be found in *thetests/L Tlib.c* file of the *lsof* distribution. The library uses the first aid, *thelsof_fields.h* header file.

BLOCKS AND TIMEOUTS

Lsof can be blocked by some kernel functions that it uses – *lstat(2)*, *readlink(2)*, and *stat(2)*. These functions are stalled in the kernel, for example, when the hosts where mounted NFS file systems reside become inaccessible.

Lsof attempts to break these blocks with timers and child processes, but the techniques are not wholly reliable. When *lsof* does manage to break a block, it will report the break with an error message. The messages may be suppressed with the **-t** and **-w** options.

The default timeout value may be displayed with the **-h** or **-?** option, and it may be changed with the **-S [t]** option. The minimum for *t* is two seconds, but you should avoid small values, since slow system responsiveness can cause short timeouts to expire unexpectedly and perhaps stop *lsof* before it can produce any output.

When *lsof* has to break a block during its access of mounted file system information, it normally continues, although with less information available to display about open files.

Lsof can also be directed to avoid the protection of timers and child processes when using the kernel functions that might block by specifying the **-O** option. While this will allow *lsof* to start up with less overhead, it exposes *lsof* completely to the kernel situations that might block it. Use this option cautiously.

AVOIDING KERNEL BLOCKS

You can use the **-b** option to tell *lsof* to avoid using kernel functions that would block. Some cautions apply.

First, using this option usually requires that your system supply alternate device numbers in place of the device numbers that *lsof* would normally obtain with the *lstat(2)* and *stat(2)* kernel functions. See the **ALTERNATE DEVICE NUMBERS** section for more information on alternate device numbers.

Second, you can't specify *names* for *lsof* to locate unless they're file system names. This is because *lsof* needs to know the device and inode numbers of files listed with *names* in the *lsof* options, and the **-b** option prevents *lsof* from obtaining them. Moreover, since *lsof* only has device numbers for the file systems that have alternates, its ability to locate files on file systems depends completely on the availability and accuracy of the alternates. If no alternates are available, or if they're incorrect, *lsof* won't be able to locate files on the named file systems.

Third, if the names of your file system directories that *lsof* obtains from your system's mount table are symbolic links, *lsof* won't be able to resolve the links. This is because the **-b** option causes *lsof* to avoid the kernel *readlink(2)* function it uses to resolve symbolic links.

Finally, using the **-b** option causes *lsof* to issue warning messages when it needs to use the kernel functions that the **-b** option directs it to avoid. You can suppress these messages by specifying the **-w** option, but if you do, you won't see the alternate device numbers reported in the warning messages.

ALTERNATE DEVICE NUMBERS

On some dialects, when *lsof* has to break a block because it can't get information about a mounted file system via the *lstat(2)* and *stat(2)* kernel functions, or because you specified the **-b** option, *lsof* can obtain some of the information it needs – the device number and possibly the file system type – from the system mount table. When that is possible, *lsof* will report the device number it obtained. (You can suppress the report by specifying the **-w** option.)

You can assist this process if your mount table is supported with an */etc/mtab* or */etc/mnttab* file that contains an options field by adding a “*dev=xxxx*” field for mount points that do not have one in their options strings. Note: you must be able to edit the file – i.e., some mount tables like recent Solaris */etc/mnttab* or Linux */proc/mounts* are read-only and can't be modified.

You may also be able to supply device numbers using the **+m** and **+m m** options, provided they are supported by your dialect. Check the output of *lsof*'s **-h** or **-?** options to see if the **+m** and **+m m** options are available.

The “*xxxx*” portion of the field is the hexadecimal value of the file system's device number. (Consult the *st_dev* field of the output of the *lstat(2)* and *stat(2)* functions for the appropriate values for your file systems.) Here's an example from a Sun Solaris 2.6 */etc/mnttab* for a file system remotely mounted via NFS:

```
nfs ignore,noquota,dev=2a40001
```

There's an advantage to having “*dev=xxxx*” entries in your mount table file, especially for file systems that are mounted from remote NFS servers. When a remote server crashes and you want to identify its users by running *lsof* on one of its clients, *lsof* probably won't be able to get output from the *lstat(2)* and *stat(2)* functions for the file system. If it can obtain the file system's device number from the mount table, it will be able to display the files open on the crashed NFS server.

Some dialects that do not use an ASCII */etc/mtab* or */etc/mnttab* file for the mount table may still provide an alternative device number in their internal mount tables. This includes AIX, Apple Darwin, FreeBSD, NetBSD, OpenBSD, and Tru64 UNIX. *Lsof* knows how to obtain the alternative device number for these dialects and uses it when its attempt to *lstat(2)* or *stat(2)* the file system is blocked.

If you're not sure your dialect supplies alternate device numbers for file systems from its mount table, use this *lsof* incantation to see if it reports any alternate device numbers:

```
lsof -b
```

Look for standard error file warning messages that begin “assuming “*dev=xxxx*” from ...”.

KERNEL NAME CACHE

Lsof is able to examine the kernel's name cache or use other kernel facilities (e.g., the ADVFS 4.x *tag_to_path()* function under Tru64 UNIX) on some dialects for most file system types, excluding AFS, and extract recently used path name components from it. (AFS file system path lookups don't use the kernel's

name cache; some Solaris VxFS file system operations apparently don't use it, either.)

lsof reports the complete paths it finds in the NAME column. If *lsof* can't report all components in a path, it reports in the NAME column the file system name, followed by a space, two '-' characters, another space, and the name components it has located, separated by the '/' character.

When *lsof* is run in repeat mode – i.e., with the **-r** option specified – the extent to which it can report path name components for the same file may vary from cycle to cycle. That's because other running processes can cause the kernel to remove entries from its name cache and replace them with others.

Lsof's use of the kernel name cache to identify the paths of files can lead it to report incorrect components under some circumstances. This can happen when the kernel name cache uses device and node number as a key (e.g., SCO OpenServer) and a key on a rapidly changing file system is reused. If the UNIX dialect's kernel doesn't purge the name cache entry for a file when it is unlinked, *lsof* may find a reference to the wrong entry in the cache. The *lsof* FAQ (The **FAQ** section gives its location.) has more information on this situation.

Lsof can report path name components for these dialects:

- FreeBSD
- HP-UX
- Linux
- NetBSD
- NEXTSTEP
- OpenBSD
- OPENSTEP
- SCO OpenServer
- SCO|Caldera UnixWare
- Solaris
- Tru64 UNIX

Lsof can't report path name components for these dialects:

- AIX

If you want to know why *lsof* can't report path name components for some dialects, see the *lsof* FAQ (The **FAQ** section gives its location.)

DEVICE CACHE FILE

Examining all members of the */dev* (or */devices*) node tree with *stat(2)* functions can be time consuming. What's more, the information that *lsof* needs – device number, inode number, and path – rarely changes.

Consequently, *lsof* normally maintains an ASCII text file of cached */dev* (or */devices*) information (exception: the */proc*-based Linux *lsof* where it's not needed.) The local system administrator who builds *lsof* can control the way the device cache file path is formed, selecting from these options:

- Path from the **-D** option;
- Path from an environment variable;
- System-wide path;
- Personal path (the default);
- Personal path, modified by an environment variable.

Consult the output of the **-h**, **-D?**, or **-?** help options for the current state of device cache support. The help output lists the default read-mode device cache file path that is in effect for the current invocation of *lsof*. The **-D?** option output lists the read-only and write device cache file paths, the names of any applicable environment variables, and the personal device cache path format.

Lsof can detect that the current device cache file has been accidentally or maliciously modified by integrity checks, including the computation and verification of a sixteen bit Cyclic Redundancy Check (CRC) sum on the file's contents. When *lsof* senses something wrong with the file, it issues a warning and attempts to remove the current cache file and create a new copy, but only to a path that the process can legitimately write.

The path from which a *lsof* process may attempt to read a device cache file may not be the same as the path to which it can legitimately write. Thus when *lsof* senses that it needs to update the device cache file, it may choose a different path for writing it from the path from which it read an incorrect or outdated version.

If available, the **-Dr** option will inhibit the writing of a new device cache file. (It's always available when specified without a path name argument.)

When a new device is added to the system, the device cache file may need to be recreated. Since *lsof* compares the mtime of the device cache file with the mtime and ctime of the */dev* (or */devices*) directory, it usually detects that a new device has been added; in that case *lsof* issues a warning message and attempts to rebuild the device cache file.

Whenever *lsof* writes a device cache file, it sets its ownership to the real UID of the executing process, and its permission modes to 0600, this restricting its reading and writing to the file's owner.

LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS

Two permissions of the *lsof* executable affect its ability to access device cache files. The permissions are set by the local system administrator when *lsof* is installed.

The first and rarer permission is setuid-root. It comes into effect when *lsof* is executed; its effective UID is then root, while its real (i.e., that of the logged-on user) UID is not. The *lsof* distribution recommends that versions for these dialects run setuid-root.

HP-UX 11.11 and 11.23
Linux

The second and more common permission is setgid. It comes into effect when the effective group IDentification number (GID) of the *lsof* process is set to one that can access kernel memory devices – e.g., “kmem”, “sys”, or “system”.

An *lsof* process that has setgid permission usually surrenders the permission after it has accessed the kernel memory devices. When it does that, *lsof* can allow more liberal device cache path formations. The *lsof* distribution recommends that versions for these dialects run setgid and be allowed to surrender setgid permission.

AIX 5.[12] and 5.3-ML1
Apple Darwin 7.x Power Macintosh systems
FreeBSD 4.x, 4.1x, 5.x and [6789].x for x86-based systems
FreeBSD 5.x, [6789].x and 1[012].8 for Alpha, AMD64 and Sparc64
based systems
HP-UX 11.00
NetBSD 1.[456], 2.x and 3.x for Alpha, x86, and SPARC-based
systems
NEXTSTEP 3.[13] for NEXTSTEP architectures
OpenBSD 2.[89] and 3.[0–9] for x86-based systems
OPENSTEP 4.x
SCO OpenServer Release 5.0.6 for x86-based systems
SCO|Caldera UnixWare 7.1.4 for x86-based systems
Solaris 2.6, 8, 9 and 10
Tru64 UNIX 5.1

(Note: *lsof* for AIX 5L and above needs setuid-root permission if its **-X** option is used.)

lsof for these dialects does not support a device cache, so the permissions given to the executable don't apply to the device cache file.

Linux

DEVICE CACHE FILE PATH FROM THE **-D OPTION**

The **-D** option provides limited means for specifying the device cache file path. Its **? function** will report the read-only and write device cache file paths that *lsof* will use.

When the **-D b, r, and u** functions are available, you can use them to request that the cache file be built in a

specific location (**b**[*path*]); read but not rebuilt (**r**[*path*]); or read and rebuilt (**u**[*path*]). The **b**, **r**, and **u** functions are restricted under some conditions. They are restricted when the *lsof* process is setuid–root. The path specified with the **r** function is always read–only, even when it is available.

The **b**, **r**, and **u** functions are also restricted when the *lsof* process runs setgid and *lsof* doesn't surrender the setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for a list of implementations that normally don't surrender their setgid permission.)

A further **-D** function, **i** (for ignore), is always available.

When available, the **b** function tells *lsof* to read device information from the kernel with the *stat*(2) function and build a device cache file at the indicated path.

When available, the **r** function tells *lsof* to read the device cache file, but not update it. When a path argument accompanies **-Dr**, it names the device cache file path. The **r** function is always available when it is specified without a path name argument. If *lsof* is not running setuid–root and surrenders its setgid permission, a path name argument may accompany the **r** function.

When available, the **u** function tells *lsof* to attempt to read and use the device cache file. If it can't read the file, or if it finds the contents of the file incorrect or outdated, it will read information from the kernel, and attempt to write an updated version of the device cache file, but only to a path it considers legitimate for the *lsof* process effective and real UIDs.

DEVICE CACHE PATH FROM AN ENVIRONMENT VARIABLE

lsof's second choice for the device cache file is the contents of the LSOFDEVCACHE environment variable. It avoids this choice if the *lsof* process is setuid–root, or the real UID of the process is root.

A further restriction applies to a device cache file path taken from the LSOFDEVCACHE environment variable: *lsof* will not write a device cache file to the path if the *lsof* process doesn't surrender its setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for information on implementations that don't surrender their setgid permission.)

The local system administrator can disable the use of the LSOFDEVCACHE environment variable or change its name when building *lsof*. Consult the output of **-D?** for the environment variable's name.

SYSTEM-WIDE DEVICE CACHE PATH

The local system administrator may choose to have a system–wide device cache file when building *lsof*. That file will generally be constructed by a special system administration procedure when the system is booted or when the contents of */dev* or */devices* changes. If defined, it is *lsof*'s third device cache file path choice.

You can tell that a system–wide device cache file is in effect for your local installation by examining the *lsof* help option output – i.e., the output from the **-h** or **-?** option.

lsof will never write to the system–wide device cache file path by default. It must be explicitly named with a **-D** function in a root–owned procedure. Once the file has been written, the procedure must change its permission modes to 0644 (owner–read and owner–write, group–read, and other–read).

PERSONAL DEVICE CACHE PATH (DEFAULT)

The default device cache file path of the *lsof* distribution is one recorded in the home directory of the real UID that executes *lsof*. Added to the home directory is a second path component of the form *lsof_hostname*.

This is *lsof*'s fourth device cache file path choice, and is usually the default. If a system–wide device cache file path was defined when *lsof* was built, this fourth choice will be applied when *lsof* can't find the system–wide device cache file. This is the **only** time *lsof* uses two paths when reading the device cache file.

The *hostname* part of the second component is the base name of the executing host, as returned by *gethostname*(2). The base name is defined to be the characters preceding the first ‘.’ in the *gethostname*(2) output, or all the *gethostname*(2) output if it contains no ‘.’.

The device cache file belongs to the user ID and is readable and writable by the user ID alone – i.e., its modes are 0600. Each distinct real user ID on a given host that executes *lsof* has a distinct device cache

file. The *hostname* part of the path distinguishes device cache files in an NFS-mounted home directory into which device cache files are written from several different hosts.

The personal device cache file path formed by this method represents a device cache file that *lsof* will attempt to read, and will attempt to write should it not exist or should its contents be incorrect or outdated.

The **-Dr** option without a path name argument will inhibit the writing of a new device cache file.

The **-D?** option will list the format specification for constructing the personal device cache file. The conversions used in the format specification are described in the *00DCACHE* file of the *lsof* distribution.

MODIFIED PERSONAL DEVICE CACHE PATH

If this option is defined by the local system administrator when *lsof* is built, the LSOPERSDCPATH environment variable contents may be used to add a component of the personal device cache file path.

The LSOPERSDCPATH variable contents are inserted in the path at the place marked by the local system administrator with the “%p” conversion in the HASPERSDC format specification of the dialect’s *machine.h* header file. (It’s placed right after the home directory in the default *lsof* distribution.)

Thus, for example, if LSOPERSDCPATH contains “LSOF”, the home directory is “/Homes/abe”, the host name is “lsof.itap.purdue.edu”, and the HASPERSDC format is the default (“%h/%p.lsof_%L”), the modified personal device cache file path is:

```
/Homes/abe/LSOF/.lsof_vic
```

The LSOPERSDCPATH environment variable is ignored when the *lsof* process is setuid-root or when the real UID of the process is root.

lsof will not write to a modified personal device cache file path if the *lsof* process doesn’t surrender setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for a list of implementations that normally don’t surrender their setgid permission.)

If, for example, you want to create a sub-directory of personal device cache file paths by using the LSOPERSDCPATH environment variable to name it, and *lsof* doesn’t surrender its setgid permission, you will have to allow *lsof* to create device cache files at the standard personal path and move them to your subdirectory with shell commands.

The local system administrator may: disable this option when *lsof* is built; change the name of the environment variable from LSOPERSDCPATH to something else; change the HASPERSDC format to include the personal path component in another place; or exclude the personal path component entirely. Consult the output of the **-D?** option for the environment variable’s name and the HASPERSDC format specification.

DIAGNOSTICS

Errors are identified with messages on the standard error file.

lsof returns a one (1) if any error was detected, including the failure to locate command names, file names, Internet addresses or files, login names, NFS files, PIDs, PGIDs, or UIDs it was asked to list. If the **-V** option is specified, *lsof* will indicate the search items it failed to list.

It returns a zero (0) if no errors were detected and if it was able to list some information about all the specified search arguments.

When *lsof* cannot open access to */dev* (or */devices*) or one of its subdirectories, or get information on a file in them with *stat(2)*, it issues a warning message and continues. That *lsof* will issue warning messages about inaccessible files in */dev* (or */devices*) is indicated in its help output – requested with the **-h** or **>B -?** options – with the message:

Inaccessible /dev warnings are enabled.

The warning message may be suppressed with the **-w** option. It may also have been suppressed by the system administrator when *lsof* was compiled by the setting of the **WARNDEVACCESS** definition. In this case, the output from the help options will include the message:

Inaccessible /dev warnings are disabled.

Inaccessible device warning messages usually disappear after *lsof* has created a working device cache file.

EXAMPLES

For a more extensive set of examples, documented more fully, see the *00QUICKSTART* file of the *lsof* distribution.

To list all open files, use:

```
lsof
```

To list all open Internet, x.25 (HP-UX), and UNIX domain files, use:

```
lsof -i -U
```

To list all open IPv4 network files in use by the process whose PID is 1234, use:

```
lsof -i 4 -a -p 1234
```

Presuming the UNIX dialect supports IPv6, to list only open IPv6 network files, use:

```
lsof -i 6
```

To list all files using any protocol on ports 513, 514, or 515 of host wonderland.cc.purdue.edu, use:

```
lsof -i @wonderland.cc.purdue.edu:513-515
```

To list all files using any protocol on any port of mace.cc.purdue.edu (cc.purdue.edu is the default domain), use:

```
lsof -i @mace
```

To list all open files for login name “abe”, or user ID 1234, or process 456, or process 123, or process 789, use:

```
lsof -p 456,123,789 -u 1234,abe
```

To list all open files on device /dev/hd4, use:

```
lsof /dev/hd4
```

To find the process that has /u/abe/foo open, use:

```
lsof /u/abe/foo
```

To send a SIGHUP to the processes that have /u/abe/bar open, use:

```
kill -HUP `lsof -t /u/abe/bar`
```

To find any open file, including an open UNIX domain socket file, with the name /dev/log, use:

```
lsof /dev/log
```

To find processes with open files on the NFS file system named /nfs/mount/point whose server is inaccessible, and presuming your mount table supplies the device number for /nfs/mount/point, use:

```
lsof -b /nfs/mount/point
```

To do the preceding search with warning messages suppressed, use:

```
lsof -bw /nfs/mount/point
```

To ignore the device cache file, use:

```
lsof -Di
```

To obtain PID and command name field output for each process, file descriptor, file device number, and file inode number for each file of each process, use:

```
lsof -FpcfDi
```

To list the files at descriptors 1 and 3 of every process running the *lsof* command for login ID “abe” every 10 seconds, use:

```
lsof -c lsof -a -d 1 -d 3 -u abe -r10
```

To list the current working directory of processes running a command that is exactly four characters long and has an 'o' or 'O' in character three, use this regular expression form of the **-c** option:

```
lsof -c ^..o./i -a -d cwd
```

To find an IP version 4 socket file by its associated numeric dot-form address, use:

```
lsof -i@128.210.15.17
```

To find an IP version 6 socket file (when the UNIX dialect supports IPv6) by its associated numeric colon-form address, use:

```
lsof -i@[0:1:2:3:4:5:6:7]
```

To find an IP version 6 socket file (when the UNIX dialect supports IPv6) by an associated numeric colon-form address that has a run of zeroes in it – e.g., the loop-back address – use:

```
lsof -i@[::1]
```

To obtain a repeat mode marker line that contains the current time, use:

```
lsof -rm====%T=====
```

To add spaces to the previous marker line, use:

```
lsof -r "m==== %T ====="
```

BUGS

Since *lsof* reads kernel memory in its search for open files, rapid changes in kernel memory may produce unpredictable results.

When a file has multiple record locks, the lock status character (following the file descriptor) is derived from a test of the first lock structure, not from any combination of the individual record locks that might be described by multiple lock structures.

Lsof can't search for files with restrictive access permissions by *name* unless it is installed with root set-UID permission. Otherwise it is limited to searching for files to which its user or its set-GID group (if any) has access permission.

The display of the destination address of a raw socket (e.g., for *ping*) depends on the UNIX operating system. Some dialects store the destination address in the raw socket's protocol control block, some do not.

Lsof can't always represent Solaris device numbers in the same way that *ls(1)* does. For example, the major and minor device numbers that the *lstat(2)* and *stat(2)* functions report for the directory on which CD-ROM files are mounted (typically */cdrom*) are not the same as the ones that it reports for the device on which CD-ROM files are mounted (typically */dev/sr0*). (*Lsof* reports the directory numbers.)

The support for */proc* file systems is available only for BSD and Tru64 UNIX dialects, Linux, and dialects derived from SYSV R4 – e.g., FreeBSD, NetBSD, OpenBSD, Solaris, UnixWare.

Some */proc* file items – device number, inode number, and file size – are unavailable in some dialects. Searching for files in a */proc* file system may require that the full path name be specified.

No text (**txt**) file descriptors are displayed for Linux processes. All entries for files other than the current working directory, the root directory, and numerical file descriptors are labeled **mem** descriptors.

Lsof can't search for Tru64 UNIX named pipes by name, because their kernel implementation of *Istat(2)* returns an improper device number for a named pipe.

Lsof can't report fully or correctly on HP-UX 9.01, 10.20, and 11.00 locks because of insufficient access to kernel data or errors in the kernel data. See the *lsof* FAQ (The **FAQ** section gives its location.) for details.

The AIX SMT file type is a fabrication. It's made up for file structures whose type (15) isn't defined in the AIX */usr/include/sys/file.h* header file. One way to create such file structures is to run X clients with the DISPLAY variable set to “:0.0”.

The `+|-f[cfn]` option is not supported under /proc-based Linux *lsof*, because it doesn't read kernel structures from kernel memory.

ENVIRONMENT

lsof may access these environment variables.

LANG	defines a language locale. See <i>setlocale(3)</i> for the names of other variables that can be used in place of LANG – e.g., LC_ALL, LC_TYPE, etc.
LSOFDEVCACHE	defines the path to a device cache file. See the DEVICE CACHE PATH FROM AN ENVIRONMENT VARIABLE section for more information.
LSOFPERSDCPATH	defines the middle component of a modified personal device cache file path. See the MODIFIED PERSONAL DEVICE CACHE PATH section for more information.

FAQ

Frequently-asked questions and their answers (an FAQ) are available in the *00FAQ* file of the *lsof* distribution.

That file is also available via anonymous ftp from *lsof.itap.purdue.edu* at *pub/tools/unix/lsofFAQ*. The URL is:

<ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/FAQ>

FILES

<i>/dev/kmem</i>	kernel virtual memory device
<i>/dev/mem</i>	physical memory device
<i>/dev/swap</i>	system paging device
<i>.lsof_hostname</i>	<i>lsof</i> 's device cache file (The suffix, <i>hostname</i> , is the first component of the host's name returned by <i>gethostname(2)</i> .)

AUTHORS

lsof was written by Victor A. Abell <abe@purdue.edu> of Purdue University. Many others have contributed to *lsof*. They're listed in the *00CREDITS* file of the *lsof* distribution.

DISTRIBUTION

The latest distribution of *lsof* is available via anonymous ftp from the host *lsof.itap.purdue.edu*. You'll find the *lsof* distribution in the *pub/tools/unix/lsof* directory.

You can also use this URL:

<ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof>

lsof is also mirrored elsewhere. When you access *lsof.itap.purdue.edu* and change to its *pub/tools/unix/lsof* directory, you'll be given a list of some mirror sites. The *pub/tools/unix/lsof* directory also contains a more complete list in its *mirrors* file. Use mirrors with caution – not all mirrors always have the latest *lsof* revision.

Some pre-compiled *lsof* executables are available on *lsof.itap.purdue.edu*, but their use is discouraged – it's better that you build your own from the sources. If you feel you must use a pre-compiled executable, please read the cautions that appear in the *README* files of the *pub/tools/unix/lsof/binaries* subdirectories and in the *00** files of the distribution.

More information on the *lsof* distribution can be found in its *README.lsof_<version>* file. If you intend to get the *lsof* distribution and build it, please read *README.lsof_<version>* and the other *00** files of the distribution before sending questions to the author.

SEE ALSO

Not all the following manual pages may exist in every UNIX dialect to which *lsof* has been ported.

access(2), *awk(1)*, *crash(1)*, *fattach(3C)*, *ff(1)*, *fstat(8)*, *fuser(1)*, *gethostname(2)*, *isprint(3)*, *kill(1)*, *localtime(3)*, *lstat(2)*, *modload(8)*, *mount(8)*, *netstat(1)*, *ofiles(8L)*, *perl(1)*, *ps(1)*, *readlink(2)*, *setlocale(3)*,

stat(2), strftime(3), time(2), uname(1).

NAME

lspci – list all PCI devices

SYNOPSIS

lspci [options]

DESCRIPTION

lspci is a utility for displaying information about PCI buses in the system and devices connected to them.

By default, it shows a brief list of devices. Use the options described below to request either a more verbose output or output intended for parsing by other programs.

If you are going to report bugs in PCI device drivers or in *lspci* itself, please include output of "lspci -vvx" or even better "lspci -vvvxx" (however, see below for possible caveats).

Some parts of the output, especially in the highly verbose modes, are probably intelligible only to experienced PCI hackers. For exact definitions of the fields, please consult either the PCI specifications or the **header.h** and **/usr/include/linux/pci.h** include files.

Access to some parts of the PCI configuration space is restricted to root on many operating systems, so the features of *lspci* available to normal users are limited. However, *lspci* tries its best to display as much as available and mark all other information with <access denied> text.

OPTIONS

Basic display modes

- m Dump PCI device data in a backward-compatible machine readable form. See below for details.
- mm Dump PCI device data in a machine readable form for easy parsing by scripts. See below for details.
- t Show a tree-like diagram containing all buses, bridges, devices and connections between them.

Display options

- v Be verbose and display detailed information about all devices.
- vv Be very verbose and display more details. This level includes everything deemed useful.
- vvv Be even more verbose and display everything we are able to parse, even if it doesn't look interesting at all (e.g., undefined memory regions).
- k Show kernel drivers handling each device and also kernel modules capable of handling it. Turned on by default when -v is given in the normal mode of output. (Currently works only on Linux with kernel 2.6 or newer.)
- x Show hexadecimal dump of the standard part of the configuration space (the first 64 bytes or 128 bytes for CardBus bridges).
- xxx Show hexadecimal dump of the whole PCI configuration space. It is available only to root as several PCI devices **crash** when you try to read some parts of the config space (this behavior probably doesn't violate the PCI standard, but it's at least very stupid). However, such devices are rare, so you needn't worry much.
- xxxx Show hexadecimal dump of the extended (4096-byte) PCI configuration space available on PCI-X 2.0 and PCI Express buses.
- b Bus-centric view. Show all IRQ numbers and addresses as seen by the cards on the PCI bus instead of as seen by the kernel.
- D Always show PCI domain numbers. By default, *lspci* suppresses them on machines which have only domain 0.

- P** Identify PCI devices by path through each bridge, instead of by bus number.
- PP** Identify PCI devices by path through each bridge, showing the bus number as well as the device number.

Options to control resolving ID's to names

- n** Show PCI vendor and device codes as numbers instead of looking them up in the PCI ID list.
- nn** Show PCI vendor and device codes as both numbers and names.
- q** Use DNS to query the central PCI ID database if a device is not found in the local **pci.ids** file. If the DNS query succeeds, the result is cached in **~/.pcids-cache** and it is recognized in subsequent runs even if **-q** is not given any more. Please use this switch inside automated scripts only with caution to avoid overloading the database servers.
- qq** Same as **-q**, but the local cache is reset.
- Q** Query the central database even for entries which are recognized locally. Use this if you suspect that the displayed entry is wrong.

Options for selection of devices

-s [[[<domain>]:]<bus>]:]<device>][.[<func>]]

Show only devices in the specified domain (in case your machine has several host bridges, they can either share a common bus number space or each of them can address a PCI domain of its own; domains are numbered from 0 to ffff), bus (0 to ff), device (0 to 1f) and function (0 to 7). Each component of the device address can be omitted or set to "*", both meaning "any value". All numbers are hexadecimal. E.g., "0:" means all devices on bus 0, "0" means all functions of device 0 on any bus, "0.3" selects third function of device 0 on all buses and ".4" shows only the fourth function of each device.

-d [<vendor>]:[<device>][:<class>]

Show only devices with specified vendor, device and class ID. The ID's are given in hexadecimal and may be omitted or given as "*", both meaning "any value".

Other options

-i <file>

Use **<file>** as the PCI ID list instead of /usr/share/misc/pci.ids.

-p <file>

Use **<file>** as the map of PCI ID's handled by kernel modules. By default, lspci uses /lib/modules/kernel_version/modules.pcimap. Applies only to Linux systems with recent enough module tools.

-M

Invoke bus mapping mode which performs a thorough scan of all PCI devices, including those behind misconfigured bridges, etc. This option gives meaningful results only with a direct hardware access mode, which usually requires root privileges. Please note that the bus mapper only scans PCI domain 0.

--version

Shows *lspci* version. This option should be used stand-alone.

PCI access options

The PCI utilities use the PCI library to talk to PCI devices (see **pcilib(7)** for details). You can use the following options to influence its behavior:

-A <method>

The library supports a variety of methods to access the PCI hardware. By default, it uses the first access method available, but you can use this option to override this decision. See **-A help** for a list of available methods and their descriptions.

-O <param>=<value>

The behavior of the library is controlled by several named parameters. This option allows one to set the value of any of the parameters. Use **-O help** for a list of known parameters and their default values.

-H1 Use direct hardware access via Intel configuration mechanism 1. (This is a shorthand for **-A intel-conf1**.)

-H2 Use direct hardware access via Intel configuration mechanism 2. (This is a shorthand for **-A intel-conf2**.)

-F <file>

Instead of accessing real hardware, read the list of devices and values of their configuration registers from the given file produced by an earlier run of lspci -x. This is very useful for analysis of user-supplied bug reports, because you can display the hardware configuration in any way you want without disturbing the user with requests for more dumps.

-G Increase debug level of the library.

MACHINE READABLE OUTPUT

If you intend to process the output of lspci automatically, please use one of the machine-readable output formats (**-m**, **-vm**, **-vmm**) described in this section. All other formats are likely to change between versions of lspci.

All numbers are always printed in hexadecimal. If you want to process numeric ID's instead of names, please add the **-n** switch.

Simple format (-m)

In the simple format, each device is described on a single line, which is formatted as parameters suitable for passing to a shell script, i.e., values separated by whitespaces, quoted and escaped if necessary. Some of the arguments are positional: slot, class, vendor name, device name, subsystem vendor name and subsystem name (the last two are empty if the device has no subsystem); the remaining arguments are option-like:

-rev Revision number.

-p *progif*

Programming interface.

The relative order of positional arguments and options is undefined. New options can be added in future versions, but they will always have a single argument not separated from the option by any spaces, so they can be easily ignored if not recognized.

Verbose format (-vmm)

The verbose output is a sequence of records separated by blank lines. Each record describes a single device by a sequence of lines, each line containing a single ‘tag: value’ pair. The *tag* and the *value* are separated by a single tab character. Neither the records nor the lines within a record are in any particular order. Tags are case-sensitive.

The following tags are defined:

Slot The name of the slot where the device resides (*[domain:]bus:device.function*). This tag is always the first in a record.

Class Name of the class.

Vendor

Name of the vendor.

Device Name of the device.

SVendor

Name of the subsystem vendor (optional).

SDevice

Name of the subsystem (optional).

PhySlot

The physical slot where the device resides (optional, Linux only).

Rev Revision number (optional).

ProgIf Programming interface (optional).

Driver Kernel driver currently handling the device (optional, Linux only).

Module

Kernel module reporting that it is capable of handling the device (optional, Linux only). Multiple lines with this tag can occur.

NUMANode

NUMA node this device is connected to (optional, Linux only).

IOMMUGroup

IOMMU group that this device is part of (optional, Linux only).

New tags can be added in future versions, so you should silently ignore any tags you don't recognize.

Backward-compatible verbose format (-vm)

In this mode, lspci tries to be perfectly compatible with its old versions. It's almost the same as the regular verbose format, but the **Device** tag is used for both the slot and the device name, so it occurs twice in a single record. Please avoid using this format in any new code.

FILES

/usr/share/misc/pci.ids

A list of all known PCI ID's (vendors, devices, classes and subclasses). Maintained at <https://pci-ids.ucw.cz/>, use the **update-pciids** utility to download the most recent version.

/usr/share/misc/pci.ids.gz

If lspci is compiled with support for compression, this file is tried before pci.ids.

~/.pciids-cache

All ID's found in the DNS query mode are cached in this file.

BUGS

Sometimes, lspci is not able to decode the configuration registers completely. This usually happens when not enough documentation was available to the authors. In such cases, it at least prints the <?> mark to signal that there is potentially something more to say. If you know the details, patches will be of course welcome.

Access to the extended configuration space is currently supported only by the **linux_sysfs** back-end.

SEE ALSO

setpci(8), **pci.ids(5)**, **update-pciids(8)**, **pcilib(7)**

AUTHOR

The PCI Utilities are maintained by Martin Mares <mj@ucw.cz>.

NAME

stat, fstat, lstat, fstatat – get file status

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/stat.h>
int stat(const char *restrict pathname,
         struct stat *restrict statbuf);
int fstat(int fd, struct stat *statbuf);
int lstat(const char *restrict pathname,
          struct stat *restrict statbuf);

#include <fcntl.h>      /* Definition of AT_* constants */
#include <sys/stat.h>

int fstatat(int dirfd, const char *restrict pathname,
            struct stat *restrict statbuf, int flags);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
lstat():
/* Since glibc 2.20 */ __DEFAULT_SOURCE
|| __XOPEN_SOURCE >= 500
|| /* Since glibc 2.10: */ __POSIX_C_SOURCE >= 200112L
|| /* glibc 2.19 and earlier */ __BSD_SOURCE

fstatat():
Since glibc 2.10:
__POSIX_C_SOURCE >= 200809L
Before glibc 2.10:
__ATFILE_SOURCE
```

DESCRIPTION

These functions return information about a file, in the buffer pointed to by *statbuf*. No permissions are required on the file itself, but—in the case of **stat()**, **fstat()**, and **lstat()**—execute (search) permission is required on all of the directories in *pathname* that lead to the file.

stat() and **fstat()** retrieve information about the file pointed to by *pathname*; the differences for **fstatat()** are described below.

lstat() is identical to **stat()**, except that if *pathname* is a symbolic link, then it returns information about the link itself, not the file that the link refers to.

fstat() is identical to **stat()**, except that the file about which information is to be retrieved is specified by the file descriptor *fd*.

The stat structure

All of these system calls return a *stat* structure (see **stat(3type)**).

Note: for performance and simplicity reasons, different fields in the *stat* structure may contain state information from different moments during the execution of the system call. For example, if *st_mode* or *st_uid* is changed by another process by calling **chmod(2)** or **chown(2)**, **stat()** might return the old *st_mode* together with the new *st_uid*, or the old *st_uid* together with the new *st_mode*.

fstatat()

The **fstatat()** system call is a more general interface for accessing file information which can still provide exactly the behavior of each of **stat()**, **lstat()**, and **fstat()**.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **stat()** and **lstat()** for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **stat()** and **lstat()**).

If *pathname* is absolute, then *dirfd* is ignored.

flags can either be 0, or include one or more of the following flags ORed:

AT_EMPTY_PATH (since Linux 2.6.39)

If *pathname* is an empty string, operate on the file referred to by *dirfd* (which may have been obtained using the **open(2)** **O_PATH** flag). In this case, *dirfd* can refer to any type of file, not just a directory, and the behavior of **fstatat()** is similar to that of **fstat()**. If *dirfd* is **AT_FDCWD**, the call operates on the current working directory. This flag is Linux-specific; define **_GNU_SOURCE** to obtain its definition.

AT_NO_AUTOMOUNT (since Linux 2.6.38)

Don't automount the terminal ("basename") component of *pathname*. Since Linux 3.1 this flag is ignored. Since Linux 4.11 this flag is implied.

AT_SYMLINK_NOFOLLOW

If *pathname* is a symbolic link, do not dereference it: instead return information about the link itself, like **lstat()**. (By default, **fstatat()** dereferences symbolic links, like **stat()**.)

See **openat(2)** for an explanation of the need for **fstatat()**.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EACCES

Search permission is denied for one of the directories in the path prefix of *pathname*. (See also **path_resolution(7)**.)

EBADF

fd is not a valid open file descriptor.

EBADF

(**fstatat()**) *pathname* is relative but *dirfd* is neither **AT_FDCWD** nor a valid file descriptor.

EFAULT

Bad address.

EINVAL

(**fstatat()**) Invalid flag specified in *flags*.

ELOOP

Too many symbolic links encountered while traversing the path.

ENAMETOOLONG

pathname is too long.

ENOENT

A component of *pathname* does not exist or is a dangling symbolic link.

ENOENT

pathname is an empty string and **AT_EMPTY_PATH** was not specified in *flags*.

ENOMEM

Out of memory (i.e., kernel memory).

ENOTDIR

A component of the path prefix of *pathname* is not a directory.

ENOTDIR

(**fstatat()**) *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

EOVERFLOW

pathname or *fd* refers to a file whose size, inode number, or number of blocks cannot be represented in, respectively, the types *off_t*, *ino_t*, or *blkcnt_t*. This error can occur when, for example, an application compiled on a 32-bit platform without *-D_FILE_OFFSET_BITS=64* calls **stat()** on a file whose size exceeds $(1 << 31) - 1$ bytes.

VERSIONS

fstatat() was added in Linux 2.6.16; library support was added in glibc 2.4.

STANDARDS

stat(), **fstat()**, **lstat()**: SVr4, 4.3BSD, POSIX.1-2001, POSIX.1-2008.

fstatat(): POSIX.1-2008.

According to POSIX.1-2001, **lstat()** on a symbolic link need return valid information only in the *st_size* field and the file type of the *st_mode* field of the *stat* structure. POSIX.1-2008 tightens the specification, requiring **lstat()** to return valid information in all fields except the mode bits in *st_mode*.

Use of the *st_blocks* and *st_blksize* fields may be less portable. (They were introduced in BSD. The interpretation differs between systems, and possibly on a single system when NFS mounts are involved.)

NOTES**C library/kernel differences**

Over time, increases in the size of the *stat* structure have led to three successive versions of **stat()**: *sys_stat()* (slot *__NR_oldstat*), *sys_newstat()* (slot *__NR_stat*), and *sys_stat64()* (slot *__NR_stat64*) on 32-bit platforms such as i386. The first two versions were already present in Linux 1.0 (albeit with different names); the last was added in Linux 2.4. Similar remarks apply for **fstat()** and **lstat()**.

The kernel-internal versions of the *stat* structure dealt with by the different versions are, respectively:

__old_kernel_stat

The original structure, with rather narrow fields, and no padding.

stat Larger *st_ino* field and padding added to various parts of the structure to allow for future expansion.

stat64 Even larger *st_ino* field, larger *st_uid* and *st_gid* fields to accommodate the Linux-2.4 expansion of UIDs and GIDs to 32 bits, and various other enlarged fields and further padding in the structure. (Various padding bytes were eventually consumed in Linux 2.6, with the advent of 32-bit device IDs and nanosecond components for the timestamp fields.)

The glibc **stat()** wrapper function hides these details from applications, invoking the most recent version of the system call provided by the kernel, and repacking the returned information if required for old binaries.

On modern 64-bit systems, life is simpler: there is a single **stat()** system call and the kernel deals with a *stat* structure that contains fields of a sufficient size.

The underlying system call employed by the glibc **fstatat()** wrapper function is actually called **fstatat64()** or, on some architectures, **newfstatat()**.

EXAMPLES

The following program calls **lstat()** and displays selected fields in the returned *stat* structure.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <time.h>

int
main(int argc, char *argv[ ])
{
```

```

    struct stat sb;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <pathname>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    if (lstat(argv[1], &sb) == -1) {
        perror("lstat");
        exit(EXIT_FAILURE);
    }

    printf("ID of containing device: [%x,%x]\n",
           major(sb.st_dev),
           minor(sb.st_dev));

    printf("File type:                 ");

    switch (sb.st_mode & S_IFMT) {
    case S_IFBLK:   printf("block device\n");          break;
    case S_IFCHR:   printf("character device\n");       break;
    case S_IFDIR:   printf("directory\n");              break;
    case S_IFIFO:   printf("FIFO/pipe\n");              break;
    case S_IFLNK:   printf("symlink\n");                break;
    case S_IFREG:   printf("regular file\n");            break;
    case S_IFSOCK:  printf("socket\n");                 break;
    default:        printf("unknown?\n");               break;
    }

    printf("I-node number:             %ju\n", (uintmax_t) sb.st_ino);

    printf("Mode:                      %jo (octal)\n",
           (uintmax_t) sb.st_mode);

    printf("Link count:                %ju\n", (uintmax_t) sb.st_nlink);
    printf("Ownership:                 UID=%ju     GID=%ju\n",
           (uintmax_t) sb.st_uid, (uintmax_t) sb.st_gid);

    printf("Preferred I/O block size: %jd bytes\n",
           (intmax_t) sb.st_blksize);
    printf("File size:                  %jd bytes\n",
           (intmax_t) sb.st_size);
    printf("Blocks allocated:          %jd\n",
           (intmax_t) sb.st_blocks);

    printf("Last status change:        %s", ctime(&sb.st_ctime));
    printf("Last file access:          %s", ctime(&sb.st_atime));
    printf("Last file modification:   %s", ctime(&sb.st_mtime));

    exit(EXIT_SUCCESS);
}

```

SEE ALSO

ls(1), stat(1), access(2), chmod(2), chown(2), readlink(2), statx(2), utime(2), stat(3type), capabilities(7), inode(7), symlink(7)

NAME

lvchange – Change the attributes of logical volume(s)

SYNOPSIS

```
lvchange option_args position_args
[ option_args ]
-a|--activate y|n|ay
    --activationmode partial|degraded|complete
    --addtag Tag
    --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
-A|--autobackup y|n
    --cachemode writethrough|writeback|passthrough
    --cachepolicy String
    --cachesettings String
    --commandprofile String
    --compression y|n
    --config String
-C|--contiguous y|n
-d|--debug
    --deduplication y|n
    --deltag Tag
    --detachprofile
    --discards passdown|nopassdown|ignore
    --driverloaded y|n
    --errorwhenfull y|n
-f|--force
-h|--help
-K|--ignoreactivationskip
    --ignorelockingfailure
    --ignoremonitoring
    --lockopt String
    --longhelp
-j|--major Number
    --[raid]maxrecoveryrate Size[k|UNIT]
    --metadataprofile String
    --minor Number
    --[raid]minrecoveryrate Size[k|UNIT]
    --monitor y|n
    --nolocking
    --noudevsync
-P|--partial
-p|--permission rw|r
-M|--persistent y|n
    --poll y|n
    --profile String
-q|--quiet
-r|--readahead auto|none|Number
    --readonly
    --rebuild PV
    --refresh
    --reportformat basic|json
    --resync
-S|--select String
-k|--setactivationskip y|n
    --[raid]syncaction check|repair
```

```
--sysinit
-t|--test
-v|--verbose
--version
--[raid]writebehind Number
--[raid]writemostly PV[:t|n|y]
-y|--yes
-Z|--zero y|n
```

DESCRIPTION

lvchange changes LV attributes in the VG, changes LV activation in the kernel, and includes other utilities for LV maintenance.

USAGE

Change a general LV attribute.

For options listed in parentheses, any one is required, after which the others are optional.

lvchange

```
( -C|--contiguous y|n,
-p|--permission rw|r,
-r|--readahead auto|none|Number,
-k|--setactivationskip y|n,
-Z|--zero y|n,
-M|--persistent n,
--addtag Tag,
--deltag Tag,
--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit,
--compression y|n,
--deduplication y|n,
--detachprofile,
--metadataprofile String,
--profile String,
--errorwhenfull y|n,
--discards passdown|nopassdown|ignore,
--cachemode writethrough|writeback|passthrough,
--cachepolicy String,
--cachesettings String,
--[raid]minrecoverystart Size[k|UNIT],
--[raid]maxrecoverystart Size[k|UNIT],
--[raid]writebehind Number,
--[raid]writemostly PV[:t|n|y] )
```

VG|LV|Tag|Select ...

```
[ -a|--activate y|n|ay ]
[ --poll y|n ]
[ --monitor y|n ]
[ COMMON_OPTIONS ]
```

Resyncronize a mirror or raid LV.

Use to reset 'R' attribute on a not initially synchronized LV.

```
lvchange --resync VG|LV_mIRROR_RAID|Tag|Select ...
[ -a|--activate y|n|ay ]
[ COMMON_OPTIONS ]
```

Resynchronize or check a raid LV.

```
lvchange --syncaction check|repair VG|LV_raid|Tag|Select ...
[ COMMON_OPTIONS ]
```

Reconstruct data on specific PVs of a raid LV.

```
lvchange --rebuild PV VG|LV_raid|Tag|Select ...
[ COMMON_OPTIONS ]
```

Activate or deactivate an LV.

```
lvchange -a|--activate y|n|ay VG|LV|Tag|Select ...
[ -P|--partial ]
[ -K|--ignoreactivationskip ]
[ --activationmode partial|degraded|complete ]
[ --poll y|n ]
[ --monitor y|n ]
[ --ignorelockingfailure ]
[ --sysinit ]
[ --readonly ]
[ COMMON_OPTIONS ]
```

Reactivate an LV using the latest metadata.

```
lvchange --refresh VG|LV|Tag|Select ...
[ -P|--partial ]
[ --activationmode partial|degraded|complete ]
[ --poll y|n ]
[ --monitor y|n ]
[ COMMON_OPTIONS ]
```

Start or stop monitoring an LV from dmeventd.

```
lvchange --monitor y|n VG|LV|Tag|Select ...
[ COMMON_OPTIONS ]
```

Start or stop processing an LV conversion.

```
lvchange --poll y|n VG|LV|Tag|Select ...
[ --monitor y|n ]
[ COMMON_OPTIONS ]
```

Make the minor device number persistent for an LV.

```
lvchange -M|--persistent y --minor Number LV
[ -j|--major Number ]
[ -a|--activate y|n|ay ]
[ --poll y|n ]
[ --monitor y|n ]
[ COMMON_OPTIONS ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ -S|--select String ]
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-a|--activate y|n|ay

Change the active state of LVs. An active LV can be used through a block device, allowing data on the LV to be accessed. **y** makes LVs active, or available. **n** makes LVs inactive, or unavailable. The block device for the LV is added or removed from the system using device-mapper in the kernel. A symbolic link /dev/VGName/LVName pointing to the device node is also added/removed. All software and scripts should access the device through the symbolic link and present this as the name of the device. The location and name of the underlying device node may depend on the distribution, configuration (e.g. udev), or release version. **ay** specifies autoactivation, in which case an LV is activated only if it matches an item in lvm.conf activation/auto_activation_volume_list. If the list is not set, all LVs are considered to match, and if the list is set but empty, no LVs match. Autoactivation should be used during system boot to make it possible to select which LVs should be automatically activated by the system. See **lvmlockd(8)** for more information about activation options **ey** and **sy** for shared VGs.

--activationmode partial|degraded|complete

Determines if LV activation is allowed when PVs are missing, e.g. because of a device failure. **complete** only allows LVs with no missing PVs to be activated, and is the most restrictive mode. **degraded** allows RAID LVs with missing PVs to be activated. (This does not include the "mirror" type, see "raid1" instead.) **partial** allows any LV with missing PVs to be activated, and should only be used for recovery or repair. For default, see lvm.conf/activation_mode. See **lvmraid(7)** for more information.

--addtag Tag

Adds a tag to a PV, VG or LV. This option can be repeated to add multiple tags at once. See **lvm(8)** for information about tags.

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the

same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--cachemode writethrough|writeback|passthrough

Specifies when writes to a cache LV should be considered complete. **writeback** considers a write complete as soon as it is stored in the cache pool. **writethrough** considers a write complete only when it has been stored in both the cache pool and on the origin LV. While writethrough may be slower for writes, it is more resilient if something should happen to a device associated with the cache pool LV. With **passthrough**, all reads are served from the origin LV (all reads miss the cache) and all writes are forwarded to the origin LV; additionally, write hits cause cache block invalidates. See **lvmcache(7)** for more information.

--cachepolicy String

Specifies the cache policy for a cache LV. See **lvmcache(7)** for more information.

--cachesettings String

Specifies tunable values for a cache LV in "Key = Value" form. Repeat this option to specify multiple values. (The default values should usually be adequate.) The special string value **default** switches settings back to their default kernel values and removes them from the list of settings stored in LVM metadata. See **lvmcache(7)** for more information.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--compression y|n

Controls whether compression is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-C|--contiguous y|n

Sets or resets the contiguous allocation policy for LVs. Default is no contiguous allocation based on a next free principle. It is only possible to change a non-contiguous allocation policy to contiguous if all of the allocated physical extents in the LV are already contiguous.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--deduplication y|n

Controls whether deduplication is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--deltag Tag

Deletes a tag from a PV, VG or LV. This option can be repeated to delete multiple tags at once. See **lvm(8)** for information about tags.

--detachprofile

Detaches a metadata profile from a VG or LV. See **lvm.conf(5)** for more information about profiles.

--discards passdown|nopassdown|ignore

Specifies how the device-mapper thin pool layer in the kernel should handle discards. **ignore** causes the thin pool to ignore discards. **nopassdown** causes the thin pool to process discards itself to allow reuse of unneeded extents in the thin pool. **passdown** causes the thin pool to process discards itself (like nopassdown) and pass the discards to the underlying device. See **lvmthin(7)** for more information.

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--errorwhenfull y|n

Specifies thin pool behavior when data space is exhausted. When yes, device-mapper will immediately return an error when a thin pool is full and an I/O request requires space. When no, device-mapper will queue these I/O requests for a period of time to allow the thin pool to be extended. Errors are returned if no space is available after the timeout. (Also see dm-thin-pool kernel module option no_space_timeout.) See **lvmthin(7)** for more information.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

-K|--ignoreactivationskip

Ignore the "activation skip" LV flag during activation to allow LVs with the flag set to be activated.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--ignoremonitoring

Do not interact with dmeventd unless --monitor is specified. Do not use this if dmeventd is already monitoring a device.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-j|--major Number

Sets the major number of an LV block device.

--[raid]maxrecoveryrate Size[k|UNIT]

Sets the maximum recovery rate for a RAID LV. The rate value is an amount of data per second for each device in the array. Setting the rate to 0 means it will be unbounded. See **lvmraid(7)** for more information.

--metadataprofile String

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--minor Number

Sets the minor number of an LV block device.

--[raid]minrecoveryrate Size[k|UNIT]

Sets the minimum recovery rate for a RAID LV. The rate value is an amount of data per second for each device in the array. Setting the rate to 0 means it will be unbounded. See **lvmraid(7)** for more information.

--monitor y|n

Start (yes) or stop (no) monitoring an LV with dmeventd. dmeventd monitors kernel events for an LV, and performs automated maintenance for the LV in response to specific events. See **dmeventd(8)** for more information.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

-P|--partial

Commands will do their best to activate LVs with missing PV extents. Missing extents may be replaced with error or zero segments according to the lvm.conf missing_stripe_filler setting. Metadata may not be changed with this option.

-p|--permission rw|r

Set access permission to read only **r** or read and write **rw**.

-M|--persistent y|n

When yes, makes the specified minor number persistent.

--poll y|n

When yes, start the background transformation of an LV. An incomplete transformation, e.g. pv-move or lvconvert interrupted by reboot or crash, can be restarted from the last checkpoint with --poll y. When no, background transformation of an LV will not occur, and the transformation will not complete. It may not be appropriate to immediately poll an LV after activation, in which case --poll n can be used to defer polling until a later --poll y command.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

-r|--readahead auto|none|Number

Sets read ahead sector count of an LV. **auto** is the default which allows the kernel to choose a suitable value automatically. **none** is equivalent to zero.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--rebuild PV

Selects a PV to rebuild in a raid LV. Multiple PVs can be rebuilt by repeating this option. Use this option in place of --resync or --syncaction repair when the PVs with corrupted data are known, and their data should be reconstructed rather than reconstructing default (rotating) data. See **lvm-raid(7)** for more information.

--refresh

If the LV is active, reload its metadata. This is not necessary in normal operation, but may be useful if something has gone wrong, or if some form of manual LV sharing is being used.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--resync

Initiates mirror synchronization. Synchronization generally happens automatically, but this option forces it to run. Also see --rebuild to synchronize a specific PV. During synchronization, data is

read from the primary mirror device and copied to the others. This can take considerable time, during which the LV is without a complete redundant copy of the data. See **lvmraid(7)** for more information.

-S|—select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by **--select help** and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See **--options help** for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-k|—setactivationskip y|n

Persistently sets (yes) or clears (no) the "activation skip" flag on an LV. An LV with this flag set is not activated unless the **--ignoreactivationskip** option is used by the activation command. This flag is set by default on new thin snapshot LVs. The flag is not applied to deactivation. The current value of the flag is indicated in the lvs lv_attr bits.

--[raid]syncaction check|repair

Initiate different types of RAID synchronization. This causes the RAID LV to read all data and parity blocks in the array and check for discrepancies (mismatches between mirrors or incorrect parity values). **check** will count but not correct discrepancies. **repair** will correct discrepancies. See lvs for reporting discrepancies found or repaired.

--sysinit

Indicates that vgchange/lvchange is being invoked from early system initialisation scripts (e.g. rc.sysinit or an initrd), before writable filesystems are available. As such, some functionality needs to be disabled and this option acts as a shortcut which selects an appropriate set of options. Currently, this is equivalent to using **--ignorelockingfailure**, **--ignoremonitoring**, **--poll n**, and setting env var **LVM_SUPPRESS_LOCKING_FAILURE_MESSAGES**. vgchange/lvchange skip autoactivation, and defer to pvscan autoactivation.

-t|—test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|—verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--[raid]writebehind Number

The maximum number of outstanding writes that are allowed to devices in a RAID1 LV that is marked write-mostly. Once this value is exceeded, writes become synchronous (i.e. all writes to the constituent devices must complete before the array signals the write has completed). Setting the value to zero clears the preference and allows the system to choose the value arbitrarily.

--[raid]writemostly PV[:t|n|y]

Mark a device in a RAID1 LV as write-mostly. All reads to these drives will be avoided unless absolutely necessary. This keeps the number of I/Os to the drive to a minimum. The default behavior is to set the write-mostly attribute for the specified PV. It is also possible to remove the write-mostly flag by adding the suffix **:n** at the end of the PV name, or to toggle the value with the suffix **:t**. Repeat this option to change the attribute on multiple PVs.

-y|—yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme

caution. (For automatic no, see `-qq`.)

-Z|--zero y|n

Set zeroing mode for thin pool. Note: already provisioned blocks from pool in non-zero mode are not cleared in unwritten parts when setting `--zero y`.

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV. LV followed by `_<type>` indicates that an LV of the given type is required. (raid represents raid<N> type)

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the `--select` option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control `--units`, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Change LV permission to read-only:

lvchange -pr vg00/lvol1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvconvert – Change logical volume layout

SYNOPSIS

```
lvconvert option_args position_args
  [ option_args ]
  [ position_args ]

  --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
  -b|--background
  -H|--cache
    --cachedevice PV
    --cachemetadataformat auto|1|2
    --cachemode writethrough|writeback|passthrough
    --cachepolicy String
    --cachepool LV
    --cachesettings String
    --cachesize Size[m|UNIT]
    --cachevol LV
  -c|--chunksize Size[k|UNIT]
    --commandprofile String
    --compression y|n
    --config String
  -d|--debug
    --deduplication y|n
    --discards passdown|nopassdown|ignore
    --driverloaded y|n
  -f|--force
  -h|--help
  -i|--interval Number
    --lockopt String
    --longhelp
    --merge
    --mergemirrors
    --mergesnapshot
    --mergethin
    --metadataprofile String
    --mirrorlog core|disk
  -m|--mirrors [+|-]Number
  -n|--name String
    --nolocking
    --noudevsync
    --originname LV
    --poolmetadata LV
    --poolmetadatasize Size[m|UNIT]
    --poolmetadataspare y|n
    --profile String
  -q|--quiet
    --raidintegrity y|n
    --raidintegrityblocksize Number
    --raidintegritymode String
  -r|--readahead auto|none|Number
  -R|--regionsize Size[m|UNIT]
    --repair
    --replace PV
  -s|--snapshot
```

```
--splitcache
--splitmirrors Number
--splitsnapshot
--startpoll
--stripes Number
-I|--stripesize Size[k|UNIT]
--swapmetadata
-t|--test
-T|--thin
--thinpool LV
--trackchanges
--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool
--uncache
--usepolicies
--vdopool LV
-v|--verbose
--version
-V|--virtualsize Size[m|UNIT]
-y|--yes
-Z|--zero y|n
```

DESCRIPTION

lvconvert changes the LV type and includes utilities for LV data maintenance. The LV type controls data layout and redundancy. The LV type is also called the segment type or segtype.

To display the current LV type, run the command:

```
lvs -o name,segtype LV
```

In some cases, an LV is a single device mapper (dm) layer above physical devices. In other cases, hidden LVs (dm devices) are layered between the visible LV and physical devices. LVs in the middle layers are called sub LVs. A command run on a visible LV sometimes operates on a sub LV rather than the specified LV. In other cases, a sub LV must be specified directly on the command line.

Sub LVs can be displayed with the command:

```
lvs -a
```

The **linear** type is equivalent to the **striped** type when one stripe exists. In that case, the types can sometimes be used interchangably.

In most cases, the **mirror** type is deprecated and the **raid1** type should be used. They are both implementations of mirroring.

Striped raid types are **raid0/raid0_meta**, **raid5** (an alias for **raid5_ls**), **raid6** (an alias for **raid6_zr**) and **raid10** (an alias for **raid10_near**).

As opposed to mirroring, **raid5** and **raid6** stripe data and calculate parity blocks. The parity blocks can be used for data block recovery in case devices fail. A maximum number of one device in a **raid5** LV may fail, and two in case of **raid6**. Striped raid types typically rotate the parity and data blocks for performance reasons, thus avoiding contention on a single device. Specific arrangements of parity and data blocks (layouts) can be used to optimize I/O performance, or to convert between raid levels. See **lvmraid(7)** for more information.

Layouts of **raid5** rotating parity blocks can be: left-asymmetric (**raid5_la**), left-symmetric (**raid5_ls** with

alias raid5), right-asymmetric (raid5_ra), right-symmetric (raid5_rs) and raid5_n, which doesn't rotate parity blocks. Layouts of raid6 are: zero-restart (raid6_zr with alias raid6), next-restart (raid6_nr), and next-continue (raid6_nc).

Layouts including _n allow for conversion between raid levels (raid5_n to raid6 or raid5_n to striped/raid0/raid0_meta). Additionally, special raid6 layouts for raid level conversions between raid5 and raid6 are: raid6_ls_6, raid6_rs_6, raid6_la_6 and raid6_ra_6. Those correspond to their raid5 counterparts (e.g. raid5_rs can be directly converted to raid6_rs_6 and vice-versa).

raid10 (an alias for raid10_near) is currently limited to one data copy and even number of sub LVs. This is a mirror group layout, thus a single sub LV may fail per mirror group without data loss.

Striped raid types support converting the layout, their stripesize and their number of stripes.

The striped raid types combined with raid1 allow for conversion from linear-> striped/raid0/raid0_meta and vice-versa by e.g. linear <-> raid1 <-> raid5_n (then adding stripes) <-> striped/raid0/raid0_meta.

USAGE

Convert LV to linear.

```
lvconvert --type linear LV
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to striped.

```
lvconvert --type striped LV
[ -I|--stripesize Size[k|UNIT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ -i|--interval Number ]
[ --stripes Number ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to type mirror (also see type raid1),

```
lvconvert --type mirror LV
[ -m|--mirrors [+|-]Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ -i|--interval Number ]
[ --stripes Number ]
[ --mirrorlog core|disk ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to raid or change raid layout
(a specific raid level must be used, e.g. raid1).

```
lvconvert --type raid LV
[ -m|--mirrors [+|-]Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ -i|--interval Number ]
```

```
[ --stripes Number ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to raid1 or mirror, or change number of mirror images.

```
lvconvert -m|--mirrors [+|-]Number LV
[ -R|--regionsize Size[m|UNIT] ]
[ -i|--interval Number ]
[ --mirrorlog core|disk ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert raid LV to change number of stripe images.

```
lvconvert --stripes Number LV_raid
[ -i|--interval Number ]
[ -R|--regionsize Size[m|UNIT] ]
[ -I|--stripesize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert raid LV to change the stripe size.

```
lvconvert -I|--stripesize Size[k|UNIT] LV_raid
[ -i|--interval Number ]
[ -R|--regionsize Size[m|UNIT] ]
[ COMMON_OPTIONS ]
```

Split images from a raid1 or mirror LV and use them to create a new LV.

```
lvconvert --splitmirrors Number -n|--name LV_new LV_cache_mirror_raid1
[ COMMON_OPTIONS ]
[ PV ... ]
```

Split images from a raid1 LV and track changes to origin for later merge.

```
lvconvert --splitmirrors Number --trackchanges LV_cache_raid1
[ COMMON_OPTIONS ]
[ PV ... ]
```

Merge LV images that were split from a raid1 LV.

```
lvconvert --mergemirrors VG|LV_linear_raid|Tag ...
[ COMMON_OPTIONS ]
```

Convert LV to a thin LV, using the original LV as an external origin.

```
lvconvert --type thin --thinpool LV LV_linear_striped_thin_cache_raid
[ -T|--thin ]
[ -r|--readahead auto|none|Number ]
[ -c|--chunksize Size[k|UNIT] ]
```

```
[ -Z|--zero y|n ]
[ --originname LV_new ]
[ --poolmetadata LV ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --metadaprofile String ]
[ COMMON_OPTIONS ]
```

Attach a cache pool to an LV, converts the LV to type cache.

```
lvconvert --type cache --cachepool LV LV_linear_stripped_thinpool_vdo_vdopool_vdopooldata_raid
[ -H|--cache ]
[ -Z|--zero y|n ]
[ -r|--readahead auto|none|Number ]
[ -c|--chunksize Size[k|UNIT] ]
[ --cachemetadataformat auto|1|2 ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --poolmetadata LV ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --metadaprofile String ]
[ COMMON_OPTIONS ]
```

Attach a writecache to an LV, converts the LV to type writecache.

```
lvconvert --type writecache --cachevol LV LV_linear_stripped_raid
[ --cachesettings String ]
[ COMMON_OPTIONS ]
```

Attach a cache to an LV, converts the LV to type cache.

```
lvconvert --type cache --cachevol LV LV_linear_stripped_thinpool_raid
[ -H|--cache ]
[ -Z|--zero y|n ]
[ -c|--chunksize Size[k|UNIT] ]
[ --cachemetadataformat auto|1|2 ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --poolmetadatasize Size[m|UNIT] ]
[ COMMON_OPTIONS ]
```

Add a writecache to an LV, using a specified cache device.

```
lvconvert --type writecache --cachedevice PV LV_linear_stripped_raid
[ --cachesize Size[m|UNIT] ]
[ --cachesettings String ]
[ COMMON_OPTIONS ]
```

Add a cache to an LV, using a specified cache device.

```
lvconvert --type cache --cachedevice PV LV_linear_striped_thinpool_raid
[ -c|--chunksize Size[k|UNIT] ]
[ --cachesize Size[m|UNIT] ]
[ --cachesettings String ]
[ COMMON_OPTIONS ]
```

Convert LV to type thin-pool.

```
lvconvert --type thin-pool LV_linear_striped_cache_raid
[ -I|--stripesize Size[k|UNIT] ]
[ -r|--readahead auto|none|Number ]
[ -c|--chunksize Size[k|UNIT] ]
[ -Z|--zero y|n ]
[ --stripes Number ]
[ --discards passdown|nopassdown|ignore ]
[ --poolmetadata LV ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --metadataprofile String ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to type cache-pool.

```
lvconvert --type cache-pool LV_linear_striped_raid
[ -Z|--zero y|n ]
[ -r|--readahead auto|none|Number ]
[ -c|--chunksize Size[k|UNIT] ]
[ --cachemetadataformat auto|1|2 ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --poolmetadata LV ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --metadataprofile String ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Convert LV to type vdopool.

```
lvconvert --type vdo-pool LV_linear_striped_cache_raid
[ -n|--name LV_new ]
[ -V|--virtualsize Size[m|UNIT] ]
[ --compression y|n ]
[ --deduplication y|n ]
[ COMMON_OPTIONS ]
```

Detach a cache from an LV.

```
lvconvert --splitcache LV_thinpool_cache_cachepool_vdopool_writecache
[ --cachesettings String ]
[ COMMON_OPTIONS ]
```

Merge thin LV into its origin LV.

```
lvconvert --mergethin LV_thin ...  
[ COMMON_OPTIONS ]
```

Merge COW snapshot LV into its origin.

```
lvconvert --mergesnapshot LV_snapshot ...  
[ -i|--interval Number ]  
[ COMMON_OPTIONS ]
```

Combine a former COW snapshot (second arg) with a former origin LV (first arg) to reverse a splitsnapshot command.

```
lvconvert --type snapshot LV LV_linear_striped  
[ -s|--snapshot ]  
[ -c|--chunksize Size[k|UNIT] ]  
[ -Z|--zero y|n ]  
[ COMMON_OPTIONS ]
```

Replace failed PVs in a raid or mirror LV.

Repair a thin pool.

Repair a cache pool.

```
lvconvert --repair LV_thinpool_cache_cachepool_mirror_raid  
[ -i|--interval Number ]  
[ --usepolicies ]  
[ --poolmetadataspare y|n ]  
[ COMMON_OPTIONS ]  
[ PV ... ]
```

Replace specific PV(s) in a raid LV with another PV.

```
lvconvert --replace PV LV_raid  
[ COMMON_OPTIONS ]  
[ PV ... ]
```

Poll LV to continue conversion.

```
lvconvert --startpoll LV_mirror_raid  
[ COMMON_OPTIONS ]
```

Add or remove data integrity checksums to raid images.

```
lvconvert --raidintegrity y|n LV_raid  
[ --raidintegritymode String ]  
[ --raidintegrityblocksize Number ]  
[ COMMON_OPTIONS ]  
[ PV ... ]
```

Common options for command:

- [**-b|--background**]
- [**-f|--force**]
- [**--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit**]
- [**--noudevsync**]

Common options for lvm:

- [**-d|--debug**]
- [**-h|--help**]
- [**-q|--quiet**]
- [**-t|--test**]
- [**-v|--verbose**]
- [**-y|--yes**]
- [**--commandprofile String**]
- [**--config String**]
- [**--driverloaded y|n**]
- [**--lockopt String**]
- [**--longhelp**]
- [**--nolocking**]
- [**--profile String**]
- [**--version**]

OPTIONS

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-b|--background

If the operation requires polling, this option causes the command to return before the operation is complete, and polling is done in the background.

-H|--cache

Specifies the command is handling a cache LV or cache pool. See **--type cache** and **--type cache-pool**. See **lvmcache(7)** for more information about LVM caching.

--cachedevice PV

The name of a device to use for a cache.

--cachemetadataformat auto|1|2

Specifies the cache metadata format used by cache target.

--cachemode writethrough|writeback|passthrough

Specifies when writes to a cache LV should be considered complete. **writeback** considers a write complete as soon as it is stored in the cache pool. **writethrough** considers a write complete only when it has been stored in both the cache pool and on the origin LV. While writethrough may be slower for writes, it is more resilient if something should happen to a device associated with the cache pool LV. With **passthrough**, all reads are served from the origin LV (all reads miss the cache) and all writes are forwarded to the origin LV; additionally, write hits cause cache block invalidates. See **lvmcache(7)** for more information.

--cachepolicy String

Specifies the cache policy for a cache LV. See **lvmcache(7)** for more information.

--cachepool *LV*

The name of a cache pool.

--cachesettings *String*

Specifies tunable values for a cache LV in "Key = Value" form. Repeat this option to specify multiple values. (The default values should usually be adequate.) The special string value **default** switches settings back to their default kernel values and removes them from the list of settings stored in LVM metadata. See **lvmcache(7)** for more information.

--cachesize *Size[m|UNIT]*

The size of cache to use.

--cachevol *LV*

The name of a cache volume.

-c|--chunksizes *Size[k|UNIT]*

The size of chunks in a snapshot, cache pool or thin pool. For snapshots, the value must be a power of 2 between 4KiB and 512KiB and the default value is 4. For a cache pool the value must be between 32KiB and 1GiB and the default value is 64. For a thin pool the value must be between 64KiB and 1GiB and the default value starts with 64 and scales up to fit the pool metadata size within 128MiB, if the pool metadata size is not specified. The value must be a multiple of 64KiB. See **lvmthin(7)** and **lvmcache(7)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--compression *y|n*

Controls whether compression is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--deduplication *y|n*

Controls whether deduplication is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--discards *passdown|nopassdown|ignore*

Specifies how the device-mapper thin pool layer in the kernel should handle discards. **ignore** causes the thin pool to ignore discards. **nopassdown** causes the thin pool to process discards itself to allow reuse of unneeded extents in the thin pool. **passdown** causes the thin pool to process discards itself (like nopassdown) and pass the discards to the underlying device. See **lvmthin(7)** for more information.

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

-i|--interval *Number*

Report progress at regular intervals.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--merge

An alias for --mergethin, --mergemirrors, or --mergesnapshot, depending on the type of LV.

--mergemirrors

Merge LV images that were split from a raid1 LV. See --splitmirrors with --trackchanges.

--mergesnapshot

Merge COW snapshot LV into its origin. When merging a snapshot, if both the origin and snapshot LVs are not open, the merge will start immediately. Otherwise, the merge will start the first time either the origin or snapshot LV are activated and both are closed. Merging a snapshot into an origin that cannot be closed, for example a root filesystem, is deferred until the next time the origin volume is activated. When merging starts, the resulting LV will have the origin's name, minor number and UUID. While the merge is in progress, reads or writes to the origin appear as being directed to the snapshot being merged. When the merge finishes, the merged snapshot is removed. Multiple snapshots may be specified on the command line or a @tag may be used to specify multiple snapshots be merged to their respective origin.

--mergethin

Merge thin LV into its origin LV. The origin thin LV takes the content of the thin snapshot, and the thin snapshot LV is removed. See **lvmthin(7)** for more information.

--metadataprofile *String*

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--mirrorlog *core|disk*

Specifies the type of mirror log for LVs with the "mirror" type (does not apply to the "raid1" type.) **disk** is a persistent log and requires a small amount of storage space, usually on a separate device from the data being mirrored. **core** is not persistent; the log is kept only in memory. In this case, the mirror must be synchronized (by copying LV data from the first device to others) each time the LV is activated, e.g. after reboot. **mirrored** is a persistent log that is itself mirrored, but should be avoided. Instead, use the raid1 type for log redundancy.

-m|--mirrors [+|-]Number

Specifies the number of mirror images in addition to the original LV image, e.g. --mirrors 1 means there are two images of the data, the original and one mirror image. Optional positional PV args on the command line can specify the devices the images should be placed on. There are two mirroring implementations: "raid1" and "mirror". These are the names of the corresponding LV types, or "segment types". Use the --type option to specify which to use (raid1 is default, and mirror is legacy) Use lvm.conf global/mirror_segtpe_default and global/raid10_segtpe_default to configure the default types. The plus prefix + can be used, in which case the number is added to the current number of images, or the minus prefix - can be used, in which case the number is subtracted from the current number of images. See **lvmraid(7)** for more information.

-n|--name *String*

Specifies the name of a new LV. When unspecified, a default name of "lvol#" is generated, where # is a number generated by LVM.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--originname *LV*

Specifies the name to use for the external origin LV when converting an LV to a thin LV. The LV being converted becomes a read-only external origin with this name.

--poolmetadata *LV*

The name of a an LV to use for storing pool metadata.

--poolmetadatasize *Size[m|UNIT]*

Specifies the size of the new pool metadata LV.

--poolmetadataspare *y|n*

Enable or disable the automatic creation and management of a spare pool metadata LV in the VG. A spare metadata LV is reserved space that can be used when repairing a pool.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--raidintegrity *y|n*

Enable or disable data integrity checksums for raid images.

--raidintegrityblocksize *Number*

The block size to use for dm-integrity on raid images. The integrity block size should usually match the device logical block size, or the file system block size. It may be less than the file system block size, but not less than the device logical block size. Possible values: 512, 1024, 2048, 4096.

--raidintegritymode *String*

Use a journal (default) or bitmap for keeping integrity checksums consistent in case of a crash. The bitmap areas are recalculated after a crash, so corruption in those areas would not be detected. A journal does not have this problem. The journal mode doubles writes to storage, but can improve performance for scattered writes packed into a single journal write. bitmap mode can in theory achieve full write throughput of the device, but would not benefit from the potential scattered write optimization.

-r|--readahead *auto|none|Number*

Sets read ahead sector count of an LV. **auto** is the default which allows the kernel to choose a suitable value automatically. **none** is equivalent to zero.

-R|--regionsize *Size[m|UNIT]*

Size of each raid or mirror synchronization region. lvm.conf activation/raid_region_size can be used to configure a default.

--repair

Replace failed PVs in a raid or mirror LV, or run a repair utility on a thin pool. See **lvmraid(7)** and **lvmthin(7)** for more information.

--replace *PV*

Replace a specific PV in a raid LV with another PV. The new PV to use can be optionally specified after the LV. Multiple PVs can be replaced by repeating this option. See **lvmraid(7)** for more information.

-s|--snapshot

Combine a former COW snapshot LV with a former origin LV to reverse a previous --splitsnapshot command.

--splitcache

Separates a cache pool from a cache LV, and keeps the unused cache pool LV. Before the separation, the cache is flushed. Also see --uncache.

--splitmirrors *Number*

Splits the specified number of images from a raid1 or mirror LV and uses them to create a new LV. If --trackchanges is also specified, changes to the raid1 LV are tracked while the split LV remains detached. If --name is specified, then the images are permanently split from the original LV and changes are not tracked.

--splitsnapshot

Separates a COW snapshot from its origin LV. The LV that is split off contains the chunks that differ from the origin LV along with metadata describing them. This LV can be wiped and then destroyed with lvremove.

--startpoll

Start polling an LV to continue processing a conversion.

--stripes *Number*

Specifies the number of stripes in a striped LV. This is the number of PVs (devices) that a striped LV is spread across. Data that appears sequential in the LV is spread across multiple devices in units of the stripe size (see --stripesize). This does not apply to existing allocated space, only newly allocated space can be striped.

-I|--stripesize *Size[k|UNIT]*

The amount of data that is written to one device before moving to the next in a striped LV.

--swapmetadata

Extracts the metadata LV from a pool and replaces it with another specified LV. The extracted LV is preserved and given the name of the LV that replaced it. Use for repair only. When the metadata LV is swapped out of the pool, it can be activated directly and used with thin provisioning tools: **cache_dump(8)**, **cache_repair(8)**, **cache_restore(8)**, **thin_dump(8)**, **thin_repair(8)**, **thin_restore(8)**.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-T|--thin

Specifies the command is handling a thin LV or thin pool. See --type thin, --type thin-pool, and --virtualsize. See **lvmthin(7)** for more information about LVM thin provisioning.

--thinpool *LV*

The name of a thin pool LV.

--trackchanges

Can be used with --splitmirrors on a raid1 LV. This causes changes to the original raid1 LV to be tracked while the split images remain detached. This is a temporary state that allows the read-only detached image to be merged efficiently back into the raid1 LV later. Only the regions with changed data are resynchronized during merge. While a raid1 LV is tracking changes, operations on it are limited to merging the split image (see --mergemirrors) or permanently splitting the image (see --splitmirrors with --name).

--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool

The LV type, also known as "segment type" or "segtype". See usage descriptions for the specific ways to use these types. For more information about redundancy and performance (**raid<N>**, **mirror**, **striped**, **linear**) see **lvmraid(7)**. For thin provisioning (**thin**, **thin-pool**) see **lvmthin(7)**. For performance caching (**cache**, **cache-pool**) see **lvmcache(7)**. For copy-on-write snapshots (**snapshot**) see usage definitions. For VDO (**vdo**) see **lvmvdo(7)**. Several commands omit an explicit type option because the type is inferred from other options or shortcuts (e.g. --stripes, --mirrors, --snapshot, --virtualsize, --thin, --cache, --vdo). Use inferred types with care because it can lead to unexpected results.

--uncache

Separates a cache pool from a cache LV, and deletes the unused cache pool LV. Before the separation, the cache is flushed. Also see **--splitcache**.

--usepolicies

Perform an operation according to the policy configured in lvm.conf or a profile.

--vdopool LV

The name of a VDO pool LV. See **lvmvdo(7)** for more information about VDO usage.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-V|--virtualsize Size[m|UNIT]

The virtual size of a new thin LV. See **lvmthin(7)** for more information about LVM thin provisioning. Using virtual size (-V) and actual size (-L) together creates a sparse LV. lvm.conf global/sparse_sectype_default determines the default segment type used to create a sparse LV. Anything written to a sparse LV will be returned when reading from it. Reading from other areas of the LV will return blocks of zeros. When using a snapshot to create a sparse LV, a hidden virtual device is created using the zero target, and the LV has the suffix _vorigin. Snapshots are less efficient than thin provisioning when creating large sparse LVs (GiB).

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

-Z|--zero y|n

For snapshots, this controls zeroing of the first 4KiB of data in the snapshot. If the LV is read-only, the snapshot will not be zeroed. For thin pools, this controls zeroing of provisioned blocks. Provisioning of large zeroed chunks negatively impacts performance.

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV. LV followed by _<type> indicates that an LV of the given type is required. (raid represents raid<N> type)

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is

TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

ADVANCED USAGE

Alternate command forms, advanced command usage, and listing of all valid syntax for completeness.

Change the region size of an LV.

```
lvconvert -R|--regionsize Size[m|UNIT] LV_raid
[COMMON_OPTIONS]
```

Change the type of mirror log used by a mirror LV.

```
lvconvert --mirrorlog core|disk LV_mirror
[COMMON_OPTIONS]
[PV ...]
```

Convert LV to a thin LV, using the original LV as an external origin (infers --type thin).

```
lvconvert -T|--thin --thinpool LV LV_linear_striped_thin_cache_raid
[-r|--readahead auto|none|Number]
[-c|--chunksize Size[k|UNIT]]
[-Z|--zero y|n]
[--type thin]
[--originname LV_new]
[--poolmetadata LV]
[--poolmetadatasize Size[m|UNIT]]
[--poolmetadataspare y|n]
[--metadataprofile String]
[COMMON_OPTIONS]
```

Attach a cache pool to an LV (infers --type cache).

```
lvconvert -H|--cache --cachepool LV LV_linear_striped_thinpool_vdo_vdopool_vdopooldata_raid
[-Z|--zero y|n]
[-r|--readahead auto|none|Number]
[-c|--chunksize Size[k|UNIT]]
[--type cache]
[--cachemetadataformat auto|1|2]
[--cachemode writethrough|writeback|passthrough]
[--cachepolicy String]
[--cachesettings String]
[--poolmetadata LV]
[--poolmetadatasize Size[m|UNIT]]
[--poolmetadataspare y|n]
[--metadataprofile String]
[COMMON_OPTIONS]
```

Attach a cache to an LV, converts the LV to type cache.

```
lvconvert -H|--cache --cachevol LV LV_linear_stripped_thinpool_raid
[ -Z|--zero y|n ]
[ -c|--chunksize Size[k|UNIT] ]
[ --cachemetadataformat auto|1|2 ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --poolmetadatasize Size[m|UNIT] ]
[ COMMON_OPTIONS ]
```

Convert LV to type vdopool.

```
lvconvert --vdopool LV_linear_stripped_cache_raid
[ -n|--name LV_new ]
[ -V|--virtualsize Size[m|UNIT] ]
[ --type vdo-pool ]
[ --compression y|n ]
[ --deduplication y|n ]
[ COMMON_OPTIONS ]
```

Detach and delete a cache from an LV.

```
lvconvert --uncache LV_thinpool_cache_vdopool_writecache
[ --cachesettings String ]
[ COMMON_OPTIONS ]
```

Swap metadata LV in a thin pool or cache pool (for repair only).

```
lvconvert --swapmetadata --poolmetadata LV LV_thinpool_cachepool
[ -c|--chunksize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
```

Merge LV that was split from a mirror (variant, use **--mergemirrors**).

Merge thin LV into its origin LV (variant, use **--mergethin**).

Merge COW snapshot LV into its origin (variant, use **--mergesnapshot**).

```
lvconvert --merge VG|LV_linear_striped_snapshot_thin_raid|Tag ...
[ -i|--interval Number ]
[ COMMON_OPTIONS ]
```

Separate a COW snapshot from its origin LV.

```
lvconvert --splitsnapshot LV_snapshot
[ COMMON_OPTIONS ]
```

Combine a former COW snapshot (second arg) with a former origin LV (first arg) to reverse a splitsnapshot command.

```
lvconvert -s|--snapshot LV LV_linear_striped
[ -c|--chunksize Size[k|UNIT] ]
[ -Z|--zero y|n ]
[ --type snapshot ]
```

[COMMON_OPTIONS]

Poll LV to continue conversion (also see **--startpoll**)
or waits till conversion/mirror syncing is finished

lvconvert *LV_mirror_raid*
[COMMON_OPTIONS]

NOTES

This previous command syntax would perform two different operations:

lvconvert --thinpool *LV1* **--poolmetadata** *LV2*

If *LV1* was not a thin pool, the command would convert *LV1* to a thin pool, optionally using a specified LV for metadata. But, if *LV1* was already a thin pool, the command would swap the current metadata LV with *LV2* (for repair purposes.)

In the same way, this previous command syntax would perform two different operations:

lvconvert --cachepool *LV1* **--poolmetadata** *LV2*

If *LV1* was not a cache pool, the command would convert *LV1* to a cache pool, optionally using a specified LV for metadata. But, if *LV1* was already a cache pool, the command would swap the current metadata LV with *LV2* (for repair purposes.)

EXAMPLES

Convert a linear LV to a two-way mirror LV.

lvconvert --type mirror --mirrors 1 *vg/lvol1*

Convert a linear LV to a two-way RAID1 LV.

lvconvert --type raid1 --mirrors 1 *vg/lvol1*

Convert a mirror LV to use an in-memory log.

lvconvert --mirrorlog core *vg/lvol1*

Convert a mirror LV to use a disk log.

lvconvert --mirrorlog disk *vg/lvol1*

Convert a mirror or raid1 LV to a linear LV.

lvconvert --type linear *vg/lvol1*

Convert a mirror LV to a raid1 LV with the same number of images.

lvconvert --type raid1 *vg/lvol1*

Convert a linear LV to a two-way mirror LV, allocating new extents from specific PV ranges.

lvconvert --mirrors 1 *vg/lvol1 /dev/sda:0-15 /dev/sdb:0-15*

Convert a mirror LV to a linear LV, freeing physical extents from a specific PV.

lvconvert --type linear *vg/lvol1 /dev/sda*

Split one image from a mirror or raid1 LV, making it a new LV.

lvconvert --splitmirrors 1 --name *lv_split* *vg/lvol1*

Split one image from a raid1 LV, and track changes made to the raid1 LV while the split image remains detached.

lvconvert --splitmirrors 1 --trackchanges *vg/lvol1*

Merge an image (that was previously created with --splitmirrors and --trackchanges) back into the original raid1 LV.

lvconvert --mergemirrors vg/lvol1_rimage_1

Replace PV /dev/sdb1 with PV /dev/sdf1 in a raid1/4/5/6/10 LV.

lvconvert --replace /dev/sdb1 vg/lvol1 /dev/sdf1

Replace 3 PVs /dev/sd[b-d]1 with PVs /dev/sd[f-h]1 in a raid1 LV.

**lvconvert --replace /dev/sdb1 --replace /dev/sdc1 --replace /dev/sdd1
vg/lvol1 /dev/sd[fgh]1**

Replace the maximum of 2 PVs /dev/sd[bc]1 with PVs /dev/sd[gh]1 in a raid6 LV.

lvconvert --replace /dev/sdb1 --replace /dev/sdc1 vg/lvol1 /dev/sd[gh]1

Convert an LV into a thin LV in the specified thin pool. The existing LV is used as an external read-only origin for the new thin LV.

lvconvert --type thin --thinpool vg/tpool1 vg/lvol1

Convert an LV into a thin LV in the specified thin pool. The existing LV is used as an external read-only origin for the new thin LV, and is renamed "external".

**lvconvert --type thin --thinpool vg/tpool1
--originname external vg/lvol1**

Convert an LV to a cache pool LV using another specified LV for cache pool metadata.

lvconvert --type cache-pool --poolmetadata vg/poolmeta1 vg/lvol1

Convert an LV to a cache LV using the specified cache pool and chunk size.

lvconvert --type cache --cachepool vg/cpool1 -c 128 vg/lvol1

Detach and keep the cache pool from a cache LV.

lvconvert --splitcache vg/lvol1

Detach and remove the cache pool from a cache LV.

lvconvert --uncache vg/lvol1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

**lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)**

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvcreate – Create a logical volume

SYNOPSIS

```
lvcreate option_args position_args
  [ option_args ]
  [ position_args ]

-a|--activate y|n|ay
  --addtag Tag
  --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
-A|--autobackup y|n
-H|--cache
  --cachedevice PV
  --cachemetadataformat auto|1|2
  --cachemode writethrough|writeback|passthrough
  --cachepolicy String
  --cachepool LV
  --cachesettings String
  --cachesize Size[m|UNIT]
  --cachevol LV
-c|--chunksize Size[k|UNIT]
  --commandprofile String
  --compression y|n
  --config String
-C|--contiguous y|n
-d|--debug
  --deduplication y|n
  --discards passdown|nopassdown|ignore
  --driverloaded y|n
  --errorwhenfull y|n
-l|--extents Number[PERCENT]
-h|--help
-K|--ignoreactivationskip
  --ignoremonitoring
  --lockopt String
  --longhelp
-j|--major Number
  --[raid]maxrecoveryrate Size[k|UNIT]
  --metadataprofile String
  --minor Number
  --[raid]minrecoveryrate Size[k|UNIT]
  --mirrorlog core|disk
-m|--mirrors Number
  --monitor y|n
-n|--name String
  --nolocking
  --nosync
  --noudevsync
-p|--permission rw|r
-M|--persistent y|n
  --poolmetadatasize Size[m|UNIT]
  --poolmetadataspare y|n
  --profile String
-q|--quiet
  --raidintegrity y|n
```

```

--raidintegrityblocksize Number
--raidintegritymode String
-r|--readahead auto|none|Number
-R|--regionsize Size[m|UNIT]
--reportformat basic|json
-k|--setactivationskip y|n
-L|--size Size[m|UNIT]
-s|--snapshot
-i|--stripes Number
-I|--stripesize Size[k|UNIT]
-t|--test
-T|--thin
--thinpool LV
--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool
--vdo
--vdopool LV
-v|--verbose
--version
-V|--virtualsize Size[m|UNIT]
-W|--wipesignatures y|n
-y|--yes
-Z|--zero y|n

```

DESCRIPTION

lvcreate creates a new LV in a VG. For standard LVs, this requires allocating logical extents from the VG's free physical extents. If there is not enough free space, the VG can be extended with other PVs (**vgextend**(8)), or existing LVs can be reduced or removed (**lvremove**(8), **lvreduce**(8).)

To control which PVs a new LV will use, specify one or more PVs as position args at the end of the command line. **lvcreate** will allocate physical extents only from the specified PVs.

lvcreate can also create snapshots of existing LVs, e.g. for backup purposes. The data in a new snapshot LV represents the content of the original LV from the time the snapshot was created.

RAID LVs can be created by specifying an LV type when creating the LV (see **lvmraid**(7)). Different RAID levels require different numbers of unique PVs be available in the VG for allocation.

Thin pools (for thin provisioning) and cache pools (for caching) are represented by special LVs with types `thin-pool` and `cache-pool` (see **lvmthin**(7) and **lvmcache**(7)). The pool LVs are not usable as standard block devices, but the LV names act as references to the pools.

Thin LVs are thinly provisioned from a thin pool, and are created with a virtual size rather than a physical size. A cache LV is the combination of a standard LV with a cache pool, used to cache active portions of the LV to improve performance.

VDO LVs are also provisioned volumes from a VDO pool, and are created with a virtual size rather than a physical size (see **lvmvdo**(7)).

Usage notes

In the usage section below, `--size Size` can be replaced with `--extents Number`. See descriptions in the options section.

In the usage section below, `--name` is omitted from the required options, even though it is typically used. When the name is not specified, a new LV name is generated with the "lvol" prefix and a unique numeric suffix.

In the usage section below, when creating a pool and the name is omitted the new LV pool name is generated with the "vpool" for vdo-pools for prefix and a unique numeric suffix.

Pool name can be specified together with VG name i.e.: vg00/mythinpool.

USAGE

Create a linear LV.

```
lvcreate -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ --type linear ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a striped LV (infers --type striped).

```
lvcreate -i|--stripes Number -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -I|--stripesize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a raid1 or mirror LV (infers --type raid1|mirror).

```
lvcreate -m|--mirrors Number -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ --mirrorlog core|disk ]
[ --[raid]minrecoveryrate Size[k|UNIT] ]
[ --[raid]maxrecoveryrate Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a raid LV (a specific raid level must be used, e.g. raid1).

```
lvcreate --type raid -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -m|--mirrors Number ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ --[raid]minrecoveryrate Size[k|UNIT] ]
[ --[raid]maxrecoveryrate Size[k|UNIT] ]
[ --raidintegrity y|n ]
[ --raidintegritymode String ]
[ --raidintegrityblocksize Number ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a raid10 LV.

```
lvcreate -m|--mirrors Number -i|--stripes Number
-L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
```

```
[ -I|--stripesize Size[k|UNIT] ]
[ -R|--regionsize Size[m|UNIT] ]
[ --[raid]minrecoverrate Size[k|UNIT] ]
[ --[raid]maxrecoverrate Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a COW snapshot LV of an origin LV.

```
lvcreate -s|--snapshot -L|--size Size[m|UNIT] LV
[ -l|--extents Number[PERCENT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ -c|--chunksize Size[k|UNIT] ]
[ --type snapshot ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin pool.

```
lvcreate --type thin-pool -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --thinpool LV_new ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a cache pool.

```
lvcreate --type cache-pool -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -H|--cache ]
[ -c|--chunksize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --cachemetadataformat auto|1|2 ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV in a thin pool (infers --type thin).

```
lvcreate -V|--virtualsize Size[m|UNIT] --thinpool LV_thinpool VG
[ -T|--thin ]
[ --type thin ]
```

```
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

Create a thin LV that is a snapshot of an existing thin LV
(infers --type thin).

```
lvcreate -s|--snapshot LV_thin
[ --type thin ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

Create a thin LV that is a snapshot of an external origin LV.

```
lvcreate --type thin --thinpool LV_thinpool LV
[ -T|--thin ]
[ -c|--chunksize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

Create a LV that returns VDO when used.

```
lvcreate --type vdo -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -V|--virtualsize Size[m|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --vdo ]
[ --vdopool LV_new ]
[ --compression y|n ]
[ --deduplication y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it,
where the new thin pool is named by the --thinpool arg.

```
lvcreate --type thin -V|--virtualsize Size[m|UNIT]
--thinpool LV_new
[ -L|--size Size[m|UNIT] --thinpool LV_new
[ -l|--extents Number[PERCENT] ]
[ -T|--thin ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

[*PV ...*]

Create a new LV, then attach the specified cachepool which converts the new LV to type cache.

```
lvcreate --type cache -L|--size Size[m|UNIT]
    --cachepool LV_cachepool VG
    [ -l|--extents Number[PERCENT] ]
    [ -H|--cache ]
    [ -c|--chunksize Size[k|UNIT] ]
    [ -i|--stripes Number ]
    [ -I|--stripesize Size[k|UNIT] ]
    [ --poolmetadatasize Size[m|UNIT] ]
    [ --poolmetadataspare y|n ]
    [ --cachemode writethrough|writeback|passthrough ]
    [ --cachepolicy String ]
    [ --cachesettings String ]
    [ --cachemetadataformat auto|1|2 ]
    [ COMMON_OPTIONS ]
    [ PV ... ]
```

Create a new LV, then attach the specified cachevol which converts the new LV to type cache.

```
lvcreate --type cache -L|--size Size[m|UNIT]
    --cachevol LV VG
    [ -l|--extents Number[PERCENT] ]
    [ -c|--chunksize Size[k|UNIT] ]
    [ -i|--stripes Number ]
    [ -I|--stripesize Size[k|UNIT] ]
    [ --cachemode writethrough|writeback|passthrough ]
    [ --cachepolicy String ]
    [ --cachesettings String ]
    [ --cachemetadataformat auto|1|2 ]
    [ COMMON_OPTIONS ]
    [ PV ... ]
```

Create a new LV, then attach a cachevol created from the specified cache device, which converts the new LV to type cache.

```
lvcreate --type cache -L|--size Size[m|UNIT]
    --cachedevice PV VG
    [ -l|--extents Number[PERCENT] ]
    [ -c|--chunksize Size[k|UNIT] ]
    [ -i|--stripes Number ]
    [ -I|--stripesize Size[k|UNIT] ]
    [ --cachemode writethrough|writeback|passthrough ]
    [ --cachepolicy String ]
    [ --cachesettings String ]
    [ --cachemetadataformat auto|1|2 ]
    [ --cachesize Size[m|UNIT] ]
    [ COMMON_OPTIONS ]
```

[*PV ...*]

Create a new LV, then attach the specified cachevol which converts the new LV to type writecache.

```
lvcREATE --type writecache -L|--size Size[m|UNIT]
    --cachevol LV VG
    [ -l|--extents Number[PERCENT] ]
    [ -i|--stripes Number ]
    [ -I|--stripesize Size[k|UNIT] ]
    [ --cachesettings String ]
    [ COMMON_OPTIONS ]
    [ PV ... ]
```

Create a new LV, then attach a cachevol created from the specified cache device, which converts the new LV to type writecache.

```
lvcREATE --type writecache -L|--size Size[m|UNIT]
    --cachedevice PV VG
    [ -l|--extents Number[PERCENT] ]
    [ -i|--stripes Number ]
    [ -I|--stripesize Size[k|UNIT] ]
    [ --cachesize Size[m|UNIT] ]
    [ --cachesettings String ]
    [ COMMON_OPTIONS ]
    [ PV ... ]
```

Common options for command:

- [-a|--activate *y|n|ay*]
- [-A|--autobackup *y|n*]
- [-C|--contiguous *y|n*]
- [-K|--ignoreactivationskip]
- [-j|--major *Number*]
- [-n|--name *String*]
- [-p|--permission *rw|r*]
- [-M|--persistent *y|n*]
- [-r|--readahead *auto|none|Number*]
- [-k|--setactivationskip *y|n*]
- [-W|--wipesignatures *y|n*]
- [-Z|--zero *y|n*]
- [--addtag *Tag*]
- [--alloc *contiguous|cling|cling_by_tags|normal|anywhere|inherit*]
- [--ignoremonitoring]
- [--metadataprofile *String*]
- [--minor *Number*]
- [--monitor *y|n*]
- [--nosync]
- [--noudevsync]
- [--reportformat *basic|json*]

Common options for lvm:

- [-d|--debug]

```
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-a|--activate y|n|ay

Controls the active state of the new LV. **y** makes the LV active, or available. New LVs are made active by default. **n** makes the LV inactive, or unavailable, only when possible. In some cases, creating an LV requires it to be active. For example, COW snapshots of an active origin LV can only be created in the active state (this does not apply to thin snapshots). The **--zero** option normally requires the LV to be active. If autoactivation **ay** is used, the LV is only activated if it matches an item in lvm.conf activation/auto_activation_volume_list. **ay** implies **--zero n** and **--wipesignatures n**. See **lvmlockd(8)** for more information about activation options for shared VGs.

--addtag Tag

Adds a tag to a PV, VG or LV. This option can be repeated to add multiple tags at once. See **lvm(8)** for information about tags.

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

-H|--cache

Specifies the command is handling a cache LV or cache pool. See **--type cache** and **--type cache-pool**. See **lvmcache(7)** for more information about LVM caching.

--cachedevice PV

The name of a device to use for a cache.

--cachemetadataformat auto|1|2

Specifies the cache metadata format used by cache target.

--cachemode writethrough|writeback|passthrough

Specifies when writes to a cache LV should be considered complete. **writeback** considers a write complete as soon as it is stored in the cache pool. **writethrough** considers a write complete only when it has been stored in both the cache pool and on the origin LV. While writethrough may be slower for writes, it is more resilient if something should happen to a device associated with the

cache pool LV. With **passthrough**, all reads are served from the origin LV (all reads miss the cache) and all writes are forwarded to the origin LV; additionally, write hits cause cache block invalidates. See **lvmcache(7)** for more information.

--cachepolicy *String*

Specifies the cache policy for a cache LV. See **lvmcache(7)** for more information.

--cachepool *LV*

The name of a cache pool.

--cachesettings *String*

Specifies tunable values for a cache LV in "Key = Value" form. Repeat this option to specify multiple values. (The default values should usually be adequate.) The special string value **default** switches settings back to their default kernel values and removes them from the list of settings stored in LVM metadata. See **lvmcache(7)** for more information.

--cachesize *Size[m|UNIT]*

The size of cache to use.

--cachevol *LV*

The name of a cache volume.

-c|--chunksize *Size[k|UNIT]*

The size of chunks in a snapshot, cache pool or thin pool. For snapshots, the value must be a power of 2 between 4KiB and 512KiB and the default value is 4. For a cache pool the value must be between 32KiB and 1GiB and the default value is 64. For a thin pool the value must be between 64KiB and 1GiB and the default value starts with 64 and scales up to fit the pool metadata size within 128MiB, if the pool metadata size is not specified. The value must be a multiple of 64KiB. See **lvmthin(7)** and **lvmcache(7)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--compression *y|n*

Controls whether compression is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-C|--contiguous *y|n*

Sets or resets the contiguous allocation policy for LVs. Default is no contiguous allocation based on a next free principle. It is only possible to change a non-contiguous allocation policy to contiguous if all of the allocated physical extents in the LV are already contiguous.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--deduplication *y|n*

Controls whether deduplication is enabled or disable for VDO volume. See **lvmvdo(7)** for more information about VDO usage.

--discards *passdown|nopassdown|ignore*

Specifies how the device-mapper thin pool layer in the kernel should handle discards. **ignore** causes the thin pool to ignore discards. **nopassdown** causes the thin pool to process discards itself to allow reuse of unneeded extents in the thin pool. **passdown** causes the thin pool to process discards itself (like nopassdown) and pass the discards to the underlying device. See **lvmthin(7)** for more information.

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--errorwhenfull y|n

Specifies thin pool behavior when data space is exhausted. When yes, device-mapper will immediately return an error when a thin pool is full and an I/O request requires space. When no, device-mapper will queue these I/O requests for a period of time to allow the thin pool to be extended. Errors are returned if no space is available after the timeout. (Also see dm-thin-pool kernel module option no_space_timeout.) See **lvmthin(7)** for more information.

-I|--extents Number[PERCENT]

Specifies the size of the new LV in logical extents. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. An alternate syntax allows the size to be determined indirectly as a percentage of the size of a related VG, LV, or set of PVs. The suffix **%VG** denotes the total size of the VG, the suffix **%FREE** the remaining free space in the VG, and the suffix **%PVS** the free space in the specified PVs. For a snapshot, the size can be expressed as a percentage of the total size of the origin LV with the suffix **%ORIGIN** (**100%ORIGIN** provides space for the whole origin). When expressed as a percentage, the size defines an upper limit for the number of logical extents in the new LV. The precise number of logical extents in the new LV is not determined until the command has completed.

-h|--help

Display help text.

-K|--ignoreactivationskip

Ignore the "activation skip" LV flag during activation to allow LVs with the flag set to be activated.

--ignoremonitoring

Do not interact with dmeventd unless --monitor is specified. Do not use this if dmeventd is already monitoring a device.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-j|--major Number

Sets the major number of an LV block device.

--[raid]maxrecoveryrate Size[k|UNIT]

Sets the maximum recovery rate for a RAID LV. The rate value is an amount of data per second for each device in the array. Setting the rate to 0 means it will be unbounded. See **lvmraid(7)** for more information.

--metadataprofile String

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--minor Number

Sets the minor number of an LV block device.

--[raid]minrecoveryrate Size[k|UNIT]

Sets the minimum recovery rate for a RAID LV. The rate value is an amount of data per second for each device in the array. Setting the rate to 0 means it will be unbounded. See **lvmraid(7)** for more information.

--mirrorlog core|disk

Specifies the type of mirror log for LVs with the "mirror" type (does not apply to the "raid1" type.) **disk** is a persistent log and requires a small amount of storage space, usually on a separate device from the data being mirrored. **core** is not persistent; the log is kept only in memory. In this case,

the mirror must be synchronized (by copying LV data from the first device to others) each time the LV is activated, e.g. after reboot. **mirrored** is a persistent log that is itself mirrored, but should be avoided. Instead, use the raid1 type for log redundancy.

-m|--mirrors Number

Specifies the number of mirror images in addition to the original LV image, e.g. --mirrors 1 means there are two images of the data, the original and one mirror image. Optional positional PV args on the command line can specify the devices the images should be placed on. There are two mirroring implementations: "raid1" and "mirror". These are the names of the corresponding LV types, or "segment types". Use the --type option to specify which to use (raid1 is default, and mirror is legacy) Use lvm.conf global/mirror_segtypes_default and global/raid10_segtypes_default to configure the default types. See the --nosync option for avoiding initial image synchronization. See **lvmraid(7)** for more information.

--monitor y|n

Start (yes) or stop (no) monitoring an LV with dmeventd. dmeventd monitors kernel events for an LV, and performs automated maintenance for the LV in response to specific events. See **dmeventd(8)** for more information.

-n|--name String

Specifies the name of a new LV. When unspecified, a default name of "lvol#" is generated, where # is a number generated by LVM.

--nolocking

Disable locking.

--nosync

Causes the creation of mirror, raid1, raid4, raid5 and raid10 to skip the initial synchronization. In case of mirror, raid1 and raid10, any data written afterwards will be mirrored, but the original contents will not be copied. In case of raid4 and raid5, no parity blocks will be written, though any data written afterwards will cause parity blocks to be stored. This is useful for skipping a potentially long and resource intensive initial sync of an empty mirror/raid1/raid4/raid5 and raid10 LV. This option is not valid for raid6, because raid6 relies on proper parity (P and Q Syndromes) being created during initial synchronization in order to reconstruct proper user data in case of device failures. raid0 and raid0_meta do not provide any data copies or parity support and thus do not support initial synchronization.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

-p|--permission rw|r

Set access permission to read only **r** or read and write **rw**.

-M|--persistent y|n

When yes, makes the specified minor number persistent.

--poolmetadatasize Size[m|UNIT]

Specifies the size of the new pool metadata LV.

--poolmetadataspare y|n

Enable or disable the automatic creation and management of a spare pool metadata LV in the VG. A spare metadata LV is reserved space that can be used when repairing a pool.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--raidintegrity y|n

Enable or disable data integrity checksums for raid images.

--raidintegrityblocksize Number

The block size to use for dm-integrity on raid images. The integrity block size should usually match the device logical block size, or the file system block size. It may be less than the file system block size, but not less than the device logical block size. Possible values: 512, 1024, 2048, 4096.

--raidintegritymode String

Use a journal (default) or bitmap for keeping integrity checksums consistent in case of a crash. The bitmap areas are recalculated after a crash, so corruption in those areas would not be detected. A journal does not have this problem. The journal mode doubles writes to storage, but can improve performance for scattered writes packed into a single journal write. bitmap mode can in theory achieve full write throughput of the device, but would not benefit from the potential scattered write optimization.

-r|--readahead auto|none|Number

Sets read ahead sector count of an LV. **auto** is the default which allows the kernel to choose a suitable value automatically. **none** is equivalent to zero.

-R|--regionsize Size[m|UNIT]

Size of each raid or mirror synchronization region. lvm.conf activation/raid_region_size can be used to configure a default.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-k|--setactivationskip y|n

Persistently sets (yes) or clears (no) the "activation skip" flag on an LV. An LV with this flag set is not activated unless the --ignoreactivationskip option is used by the activation command. This flag is set by default on new thin snapshot LVs. The flag is not applied to deactivation. The current value of the flag is indicated in the lvs lv_attr bits.

-L|--size Size[m|UNIT]

Specifies the size of the new LV. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels.

-s|--snapshot

Create a snapshot. Snapshots provide a "frozen image" of an origin LV. The snapshot LV can be used, e.g. for backups, while the origin LV continues to be used. This option can create a COW (copy on write) snapshot, or a thin snapshot (in a thin pool.) Thin snapshots are created when the origin is a thin LV and the size option is NOT specified. Thin snapshots share the same blocks in the thin pool, and do not allocate new space from the VG. Thin snapshots are created with the "activation skip" flag, see --setactivationskip. A thin snapshot of a non-thin "external origin" LV is created when a thin pool is specified. Unprovisioned blocks in the thin snapshot LV are read from the external origin LV. The external origin LV must be read-only. See **lvmthin(7)** for more information about LVM thin provisioning. COW snapshots are created when a size is specified. The size is allocated from space in the VG, and is the amount of space that can be used for saving COW blocks as writes occur to the origin or snapshot. The size chosen should depend upon the amount of writes that are expected; often 20% of the origin LV is enough. If COW space runs low, it can be extended with lvextend (shrinking is also allowed with lvreduce.) A small amount of the COW snapshot LV size is used to track COW block locations, so the full size is not available for COW data blocks. Use lvs to check how much space is used, and see --monitor to automatically extend the size to avoid running out of space.

-i|--stripes *Number*

Specifies the number of stripes in a striped LV. This is the number of PVs (devices) that a striped LV is spread across. Data that appears sequential in the LV is spread across multiple devices in units of the stripe size (see **--stripesize**). This does not change existing allocated space, but only applies to space being allocated by the command. When creating a RAID 4/5/6 LV, this number does not include the extra devices that are required for parity. The largest number depends on the RAID type (raid0: 64, raid10: 32, raid4/5: 63, raid6: 62), and when unspecified, the default depends on the RAID type (raid0: 2, raid10: 2, raid4/5: 3, raid6: 5.) To stripe a new raid LV across all PVs by default, see lvm.conf allocation/raid_stripe_all_devices.

-I|--stripesize *Size[k|UNIT]*

The amount of data that is written to one device before moving to the next in a striped LV.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-T|--thin

Specifies the command is handling a thin LV or thin pool. See **--type thin**, **--type thin-pool**, and **--virtualsize**. See **lvmthin(7)** for more information about L VM thin provisioning.

--thinpool *LV*

The name of a thin pool LV.

--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool

The LV type, also known as "segment type" or "segtype". See usage descriptions for the specific ways to use these types. For more information about redundancy and performance (**raid<N>**, **mirror**, **striped**, **linear**) see **lvmraid(7)**. For thin provisioning (**thin**, **thin-pool**) see **lvmthin(7)**. For performance caching (**cache**, **cache-pool**) see **lvmcache(7)**. For copy-on-write snapshots (**snapshot**) see usage definitions. For VDO (**vdo**) see **lvmvdo(7)**. Several commands omit an explicit type option because the type is inferred from other options or shortcuts (e.g. **--stripes**, **--mirrors**, **--snapshot**, **--virtualsize**, **--thin**, **--cache**, **--vdo**). Use inferred types with care because it can lead to unexpected results.

--vdo

Specifies the command is handling VDO LV. See **--type vdo**. See **lvmvdo(7)** for more information about VDO usage.

--vdopool *LV*

The name of a VDO pool LV. See **lvmvdo(7)** for more information about VDO usage.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-V|--virtualsize *Size[m|UNIT]*

The virtual size of a new thin LV. See **lvmthin(7)** for more information about L VM thin provisioning. Using virtual size (-V) and actual size (-L) together creates a sparse LV. lvm.conf global/sparse_segtype_default determines the default segment type used to create a sparse LV. Anything written to a sparse LV will be returned when reading from it. Reading from other areas of the LV will return blocks of zeros. When using a snapshot to create a sparse LV, a hidden virtual device is created using the zero target, and the LV has the suffix _vorigin. Snapshots are less efficient than thin provisioning when creating large sparse LVs (GiB).

-W|--wipesignatures *y|n*

Controls detection and subsequent wiping of signatures on new LVs. There is a prompt for each

signature detected to confirm its wiping (unless `--yes` is used to override confirmations.) When not specified, signatures are wiped whenever zeroing is done (see `--zero`). This behaviour can be configured with lvm.conf allocation/wipe_signatures_when_zeroing_new_lvs. If blkid wiping is used (lvm.conf allocation/use_blkid_wiping) and LVM is compiled with blkid wiping support, then the blkid(8) library is used to detect the signatures (use blkid -k to list the signatures that are recognized). Otherwise, native LVM code is used to detect signatures (only MD RAID, swap and LUKS signatures are detected in this case.) The LV is not wiped if the read only flag is set.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see `-qq`.)

-Z|--zero y|n

Controls zeroing of the first 4KiB of data in the new LV. Default is `y`. Snapshot COW volumes are always zeroed. For thin pools, this controls zeroing of provisioned blocks. LV is not zeroed if the read only flag is set. Warning: trying to mount an unzeroed LV can cause the system to hang.

VARIABLES*VG*

Volume Group name. See **[lvm\(8\)](#)** for valid names. For lvcreate, the required VG positional arg may be omitted when the VG name is included in another option, e.g. `--name VG/LV`.

LV

Logical Volume name. See **[lvm\(8\)](#)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. `VG/LV`. LV followed by `_<type>` indicates that an LV of the given type is required. (raid represents raid<N> type)

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): `PV[:PE-PE]...` Start and length range (counting from 0): `PV[:PE+PE]...`

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control `--units`, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **[lvm\(8\)](#)** for information about environment variables used by lvm. For example, `LVM_VG_NAME` can generally be substituted for a required VG parameter.

ADVANCED USAGE

Alternate command forms, advanced command usage, and listing of all valid syntax for completeness.

Create an LV that returns errors when used.

```
lvcreate --type error -L|--size Size[m|UNIT] VG
  [ -l|--extents Number[PERCENT] ]
  [ COMMON_OPTIONS ]
```

Create an LV that returns zeros when read.

```
lvcreate --type zero -L|--size Size[m|UNIT] VG
```

```
[ -l|--extents Number[PERCENT] ]
[ COMMON_OPTIONS ]
```

Create a linear LV.

```
lvcreate --type linear -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a striped LV (also see **lvcreate** --stripes).

```
lvcreate --type striped -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a mirror LV (also see --type raid1).

```
lvcreate --type mirror -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -m|--mirrors Number ]
[ -R|--regionsize Size[m|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --mirrorlog core|disk ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a COW snapshot LV of an origin LV
(also see --snapshot).

```
lvcreate --type snapshot -L|--size Size[m|UNIT] LV
[ -l|--extents Number[PERCENT] ]
[ -s|--snapshot ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ -c|--chunksize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a sparse COW snapshot LV of a virtual origin LV
(also see --snapshot).

```
lvcreate --type snapshot -L|--size Size[m|UNIT]
-V--virtualsize Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -s|--snapshot ]
[ -c|--chunksize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
```

[*PV ...*]

Create a sparse COW snapshot LV of a virtual origin LV.

```
lvcreate -s|--snapshot -L|--size Size[m|UNIT]
  -V|--virtualsize Size[m|UNIT] VG
  [ -l|--extents Number[PERCENT] ]
  [ -c|--chunksize Size[k|UNIT] ]
  [ --type snapshot ]
  [ COMMON_OPTIONS ]
  [ PV ... ]
```

Create a thin pool (infers --type thin-pool).

```
lvcreate -T|--thin -L|--size Size[m|UNIT] VG
  [ -l|--extents Number[PERCENT] ]
  [ -c|--chunksize Size[k|UNIT] ]
  [ -i|--stripes Number ]
  [ -I|--stripesize Size[k|UNIT] ]
  [ --type thin-pool ]
  [ --poolmetadatasize Size[m|UNIT] ]
  [ --poolmetadataspare y|n ]
  [ --discards passdown|nopassdown|ignore ]
  [ --errorwhenfull y|n ]
  [ COMMON_OPTIONS ]
  [ PV ... ]
```

Create a thin pool named by the --thinpool arg
(infers --type thin-pool).

```
lvcreate -L|--size Size[m|UNIT] --thinpool LV_new VG
  [ -l|--extents Number[PERCENT] ]
  [ -T|--thin ]
  [ -c|--chunksize Size[k|UNIT] ]
  [ -i|--stripes Number ]
  [ -I|--stripesize Size[k|UNIT] ]
  [ --type thin-pool ]
  [ --poolmetadatasize Size[m|UNIT] ]
  [ --poolmetadataspare y|n ]
  [ --discards passdown|nopassdown|ignore ]
  [ --errorwhenfull y|n ]
  [ COMMON_OPTIONS ]
  [ PV ... ]
```

Create a cache pool named by the --cachepool arg
(variant, uses --cachepool in place of --name).

```
lvcreate --type cache-pool -L|--size Size[m|UNIT]
  --cachepool LV_new VG
  [ -l|--extents Number[PERCENT] ]
  [ -H|--cache ]
  [ -c|--chunksize Size[k|UNIT] ]
  [ --poolmetadatasize Size[m|UNIT] ]
```

```
[ --poolmetadataspare y|n ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --cachemetadataformat auto|1|2 ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV in a thin pool.

```
lvcreate --type thin -V|--virtualsize Size[m|UNIT]
    --thinpool LV_thinpool VG
    [ -T|--thin ]
    [ -c|--chunksize Size[k|UNIT] ]
    [ --poolmetadatasize Size[m|UNIT] ]
    [ --poolmetadataspare y|n ]
    [ --discards passdown|nopassdown|ignore ]
    [ --errorwhenfull y|n ]
    [ COMMON_OPTIONS ]
```

Create a thin LV in a thin pool named in the first arg
(variant, also see --thinpool for naming pool).

```
lvcreate --type thin -V|--virtualsize Size[m|UNIT] LV_thinpool
    [ -T|--thin ]
    [ --discards passdown|nopassdown|ignore ]
    [ --errorwhenfull y|n ]
    [ COMMON_OPTIONS ]
```

Create a thin LV in the thin pool named in the first arg
(variant, infers --type thin, also see --thinpool for
naming pool.)

```
lvcreate -V|--virtualsize Size[m|UNIT] LV_thinpool
    [ -T|--thin ]
    [ --type thin ]
    [ --discards passdown|nopassdown|ignore ]
    [ --errorwhenfull y|n ]
    [ COMMON_OPTIONS ]
```

Create a thin LV that is a snapshot of an existing thin LV.

```
lvcreate --type thin LV_thin
    [ -T|--thin ]
    [ --discards passdown|nopassdown|ignore ]
    [ --errorwhenfull y|n ]
    [ COMMON_OPTIONS ]
```

Create a thin LV that is a snapshot of an existing thin LV
(infers --type thin).

```
lvcreate -T|--thin LV_thin
```

```
[ --type thin ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

Create a thin LV that is a snapshot of an external origin LV
(infers --type thin).

```
lvcreate -s|--snapshot --thinpool LV_thinpool LV
[ --type thin ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
```

Create a VDO LV with VDO pool.

```
lvcreate --vdo -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -V|--virtualsize Size[m|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --vdopool LV_new ]
[ --compression y|n ]
[ --deduplication y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a VDO LV with VDO pool.

```
lvcreate --vdopool LV_new -L|--size Size[m|UNIT] VG
[ -l|--extents Number[PERCENT] ]
[ -V|--virtualsize Size[m|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --vdo ]
[ --type vdo ]
[ --compression y|n ]
[ --deduplication y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it,
where the new thin pool is named by the --thinpool arg
(variant, infers --type thin).

```
lvcreate -V|--virtualsize Size[m|UNIT] -L|--size Size[m|UNIT]
--thinpool LV_new
[ -l|--extents Number[PERCENT] ]
[ -T|--thin ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
```

```
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it,
where the new thin pool is named by the --thinpool arg
(variant, infers --type thin).

```
lvcreate -V|--virtualsize Size[m|UNIT] -L|--size Size[m|UNIT]
--thinpool LV_new VG
[ -l|--extents Number[PERCENT] ]
[ -T|--thin ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it,
where the new thin pool is named in the first arg,
or the new thin pool name is generated when the first
arg is a VG name.

```
lvcreate --type thin -V|--virtualsize Size[m|UNIT]
--size Size[m|UNIT] VG|LV_new
[ -l|--extents Number[PERCENT] ]
[ -T|--thin ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it,
where the new thin pool is named in the first arg,
or the new thin pool name is generated when the first
arg is a VG name (variant, infers --type thin).

```
lvcreate -T|--thin -V|--virtualsize Size[m|UNIT]
--size Size[m|UNIT] VG|LV_new
[ -l|--extents Number[PERCENT] ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
```

```
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a thin LV, first creating a thin pool for it
(infers --type thin).

Create a sparse snapshot of a virtual origin LV
(infers --type snapshot).

Chooses --type thin or --type snapshot according to
config setting sparse_segtypes_default.

lvcREATE **-L|--size** *Size[m|UNIT]* **-V|--virtualsize** *Size[m|UNIT]* *VG*

```
[ -l|--extents Number[PERCENT] ]
[ -s|--snapshot ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --type snapshot ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --discards passdown|nopassdown|ignore ]
[ --errorwhenfull y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a new LV, then attach the specified cachepool
which converts the new LV to type cache
(variant, infers --type cache.)

lvcREATE **-L|--size** *Size[m|UNIT]* **--cachepool** *LV_cachepool* *VG*

```
[ -l|--extents Number[PERCENT] ]
[ -H|--cache ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --type cache ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --cachemetadataformat auto|1|2 ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Create a new LV, then attach the specified cachepool
which converts the new LV to type cache.
(variant, also use --cachepool).

lvcREATE **--type cache** **-L|--size** *Size[m|UNIT]* *LV_cachepool*

```
[ -l|--extents Number[PERCENT] ]
```

```
[ -H|--cache ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --cachemetadataformat auto|1|2 ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

When the LV arg is a cachepool, then create a new LV and attach the cachepool arg to it.

(variant, use --type cache and --cachepool.)

When the LV arg is not a cachepool, then create a new cachepool and attach it to the LV arg (alternative, use lvconvert.)

```
lvcreate -H|--cache -L|--size Size[m|UNIT] LV
[ -l|--extents Number[PERCENT] ]
[ -c|--chunksize Size[k|UNIT] ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --cachemode writethrough|writeback|passthrough ]
[ --cachepolicy String ]
[ --cachesettings String ]
[ --cachemetadataformat auto|1|2 ]
[ --poolmetadatasize Size[m|UNIT] ]
[ --poolmetadataspare y|n ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

EXAMPLES

Create a striped LV with 3 stripes, a stripe size of 8KiB and a size of 100MiB. The LV name is chosen by lvcreate.

lvcreate -i 3 -I 8 -L 100m vg00

Create a raid1 LV with two images, and a useable size of 500 MiB. This operation requires two devices, one for each mirror image. RAID metadata (superblock and bitmap) is also included on the two devices.

lvcreate --type raid1 -m1 -L 500m -n mylv vg00

Create a mirror LV with two images, and a useable size of 500 MiB. This operation requires three devices: two for mirror images and one for a disk log.

lvcreate --type mirror -m1 -L 500m -n mylv vg00

Create a mirror LV with 2 images, and a useable size of 500 MiB. This operation requires 2 devices because the log is in memory.

lvcreate --type mirror -m1 --mirrorlog core -L 500m -n mylv vg00

Create a copy-on-write snapshot of an LV:

lvcreate --snapshot --size 100m --name mysnap vg00/mylv

Create a copy-on-write snapshot with a size sufficient for overwriting 20% of the size of the original LV.
lvcreate -s -l 20%ORIGIN -n mysnap vg00/mylv

Create a sparse LV with 1TiB of virtual space, and actual space just under 100MiB.
lvcreate --snapshot --virtualsize 1t --size 100m --name mylv vg00

Create a linear LV with a usable size of 64MiB on specific physical extents.
lvcreate -L 64m -n mylv vg00 /dev/sda:0-7 /dev/sdb:0-7

Create a RAID5 LV with a usable size of 5GiB, 3 stripes, a stripe size of 64KiB, using a total of 4 devices (including one for parity).

lvcreate --type raid5 -L 5G -i 3 -I 64 -n mylv vg00

Create a RAID5 LV using all of the free space in the VG and spanning all the PVs in the VG (note that the command will fail if there are more than 8 PVs in the VG, in which case **-i 7** must be used to get to the current maximum of 8 devices including parity for RaidLVs).

**lvcreate --config allocation/raid_stripe_all_devices=1
--type raid5 -l 100%FREE -n mylv vg00**

Create RAID10 LV with a usable size of 5GiB, using 2 stripes, each on a two-image mirror. (Note that the **-i** and **-m** arguments behave differently: **-i** specifies the total number of stripes, but **-m** specifies the number of images in addition to the first image).

lvcreate --type raid10 -L 5G -i 2 -m 1 -n mylv vg00

Create a 1TiB thin LV mythin, with 256GiB thinpool tpool0 in vg00.
lvcreate --T --size 256G --name mythin vg00/tpool0

Create a 1TiB thin LV, first creating a new thin pool for it, where the thin pool has 100MiB of space, uses 2 stripes, has a 64KiB stripe size, and 256KiB chunk size.

**lvcreate --type thin --name mylv --thinpool mypool
-V 1t -L 100m -i 2 -I 64 -c 256 vg00**

Create a thin snapshot of a thin LV (the size option must not be used, otherwise a copy-on-write snapshot would be created).

lvcreate --snapshot --name mysnap vg00/thinvol

Create a thin snapshot of the read-only inactive LV named "origin" which becomes an external origin for the thin snapshot LV.

lvcreate --snapshot --name mysnap --thinpool mypool vg00/origin

Create a cache pool from a fast physical device. The cache pool can then be used to cache an LV.

lvcreate --type cache-pool -L 1G -n my_cpool vg00 /dev/fast1

Create a cache LV, first creating a new origin LV on a slow physical device, then combining the new origin LV with an existing cache pool.

**lvcreate --type cache --cachepool my_cpool
-L 100G -n mylv vg00 /dev/slow1**

Create a VDO LV vdo0 with VDOPoolLV size of 10GiB and name vpool1.

lvcreate --vdo --size 10G --name vdo0 vg00/vpool1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvccreate(8) pvdisplay(8) pvmmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvdisplay – Display information about a logical volume

SYNOPSIS

```
lvdisplay
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

lvdisplay shows the attributes of LVs, like size, read/write status, snapshot information, etc.

lvs(8) is a preferred alternative that shows the same information and more, using a more compact and configurable output format.

USAGE

```
lvdisplay
  [ -a|--all ]
  [ -c|--colon ]
  [ -C|--columns ]
  [ -H|--history ]
  [ -m|--maps ]
  [ -o|--options String ]
  [ -O|--sort String ]
  [ -S|--select String ]
  [ --aligned ]
  [ --binary ]
  [ --configreport log|vg|lv|pv|pvseg|seg ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --logonly ]
  [ --noheadings ]
  [ --nosuffix ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ --segments ]
  [ --separator String ]
  [ --shared ]
  [ --unbuffered ]
  [ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
  [ COMMON_OPTIONS ]
  [ VG|LV|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
```

[--version]

OPTIONS

--aligned

Use with --separator to align the output columns

-a|--all

Show information about internal LVs. These are components of normal LVs, such as mirrors, which are not independently accessible, e.g. not mountable.

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

-c|--colon

Generate colon separated output for easier parsing in scripts or programs. Also see **vgs(8)** which provides considerably more control over the output.

-C|--columns

Display output in columns, the equivalent of **vgs(8)**. Options listed are the same as options given in **vgs(8)**.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport log|vg|lv|pv|pvseg|seg

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

-H|--history

Include historical LVs in the output. (This has no effect unless LVs were removed while lvm.conf metadata/record_lvs_history was enabled.)

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

-m|---maps

Display the mapping of logical extents to PVs and physical extents. To map physical extents to logical extents use: pvs --segments -o+lv_name,seg_start_pe,segtype

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with --units (except h and H) if processing the output.

-o|--options String

Comma-separated, ordered list of fields to display in columns. String arg syntax is:

[+|-#]Field1[,Field2 ...] The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the -o option: -o+field1,field2 -o-field3,field4 -o#field5. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **Ivmreport(7)** for more information about reporting.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **Ivmreport(7)** for more information.

--segments**-S|--select String**

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **Ivmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator String

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when Ivmlockd is not being used on the host. See **Ivmlockd(8)** for more information about shared VGs.

-O|--sort String

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any

column with – for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-readable, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes. Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g. --units 3M.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmG-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control —units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgecfgbackup(8) vgecfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

**lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)**

**lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)**

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvextend – Add space to a logical volume

SYNOPSIS

```
lvextend option_args position_args
      [ option_args ]
      [ position_args ]

      --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
      -A|--autobackup y|n
      --commandprofile String
      --config String
      -d|--debug
      --driverloaded y|n
      -l|--extents [+]Number[PERCENT]
      -f|--force
      -h|--help
      --lockopt String
      --longhelp
      -m|--mirrors Number
      -n|--nofsck
      --nolocking
      --nosync
      --noudevsync
      --poolmetadatasize [+]Size[m|UNIT]
      --profile String
      -q|--quiet
      --reportformat basic|json
      -r|--resizefs
      -L|--size [+]Size[m|UNIT]
      -i|--stripes Number
      -I|--stripesize Size[k|UNIT]
      -t|--test
      --type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool
      --usepolicies
      -v|--verbose
      --version
      -y|--yes
```

DESCRIPTION

lvextend extends the size of an LV. This requires allocating logical extents from the VG's free physical extents. If the extension adds a new LV segment, the new segment will use the existing segment type of the LV.

Extending a copy-on-write snapshot LV adds space for COW blocks.

Use **lvconvert(8)** to change the number of data images in a RAID or mirrored LV.

In the usage section below, **--size** *Size* can be replaced with **--extents** *Number*. See both descriptions the options section.

USAGE

Extend an LV by a specified size.

```
lvextend -L|--size [+]Size[m|UNIT] LV
      [ -l|--extents [+]Number[PERCENT] ]
      [ -r|--resizefs ]
```

```
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --poolmetadatasize [+]Size[m|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Extend an LV by specified PV extents.

```
lvextend LV PV ...
[ -r|--resizesfs ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
```

Extend a pool metadata SubLV by a specified size.

```
lvextend --poolmetadatasize [+]Size[m|UNIT] LV_thinpool
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Extend an LV according to a predefined policy.

```
lvextend --usepolicies LV_snapshot_thinpool
[ -r|--resizesfs ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ -m|--mirrors Number ]
[ -n|--nofsck ]
[ --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit ]
[ --nosync ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ --type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
```

- [**--nolocking**]
- [**--profile** *String*]
- [**--version**]

OPTIONS

--alloc *contiguous|cling|cling_by_tags|normal|anywhere|inherit*

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup *y|n*

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-I|--extents [+]*Number*[**PERCENT**]

Specifies the new size of the LV in logical extents. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. An alternate syntax allows the size to be determined indirectly as a percentage of the size of a related VG, LV, or set of PVs. The suffix **%VG** denotes the total size of the VG, the suffix **%FREE** the remaining free space in the VG, and the suffix **%PVS** the free space in the specified PVs. For a snapshot, the size can be expressed as a percentage of the total size of the origin LV with the suffix **%ORIGIN** (**100%ORIGIN** provides space for the whole origin). When expressed as a percentage, the size defines an upper limit for the number of logical extents in the new LV. The precise number of logical extents in the new LV is not determined until the command has completed. When the plus + or minus - prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-m|---mirrors Number

Not used.

-n|---nofsck

Do not perform fsck before resizing filesystem when filesystem requires it. You may need to use --force to proceed with this option.

--nolocking

Disable locking.

--nosync

Causes the creation of mirror, raid1, raid4, raid5 and raid10 to skip the initial synchronization. In case of mirror, raid1 and raid10, any data written afterwards will be mirrored, but the original contents will not be copied. In case of raid4 and raid5, no parity blocks will be written, though any data written afterwards will cause parity blocks to be stored. This is useful for skipping a potentially long and resource intensive initial sync of an empty mirror/raid1/raid4/raid5 and raid10 LV. This option is not valid for raid6, because raid6 relies on proper parity (P and Q Syndromes) being created during initial synchronization in order to reconstruct proper user data in case of device failures. raid0 and raid0_meta do not provide any data copies or parity support and thus do not support initial synchronization.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--poolmetadatasize [+]*Size[m|UNIT]*

Specifies the new size of the pool metadata LV. The plus prefix+ can be used, in which case the value is added to the current size.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|---quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **[lvmreport\(7\)](#)** for more information.

-r|---resizesfs

Resize underlying filesystem together with the LV using fsadm(8).

-L|---size [+]*Size[m|UNIT]*

Specifies the new size of the LV. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. When the plus+ or minus - prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-i|---stripes Number

Specifies the number of stripes in a striped LV. This is the number of PVs (devices) that a striped LV is spread across. Data that appears sequential in the LV is spread across multiple devices in units of the stripe size (see --stripesize). This does not change existing allocated space, but only applies to space being allocated by the command. When creating a RAID 4/5/6 LV, this number does not include the extra devices that are required for parity. The largest number depends on the RAID type (raid0: 64, raid10: 32, raid4/5: 63, raid6: 62), and when unspecified, the default depends on the RAID type (raid0: 2, raid10: 2, raid4/5: 3, raid6: 5.) To stripe a new raid LV across all PVs by default, see lvm.conf allocation/raid_stripe_all_devices.

-I|--stripesize *Size[k|UNIT]*

The amount of data that is written to one device before moving to the next in a striped LV.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--type *linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool*

The LV type, also known as "segment type" or "segtype". See usage descriptions for the specific ways to use these types. For more information about redundancy and performance (**raid<N>**, **mirror**, **striped**, **linear**) see **lvmraid(7)**. For thin provisioning (**thin**, **thin-pool**) see **lvmthin(7)**. For performance caching (**cache**, **cache-pool**) see **lvmcache(7)**. For copy-on-write snapshots (**snapshot**) see usage definitions. For VDO (**vdo**) see **lvmvdo(7)**. Several commands omit an explicit type option because the type is inferred from other options or shortcuts (e.g. **--stripes**, **--mirrors**, **--snapshot**, **--virtualsize**, **--thin**, **--cache**, **--vdo**). Use inferred types with care because it can lead to unexpected results.

--usepolicies

Perform an operation according to the policy configured in lvm.conf or a profile.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

VARIABLES*LV*

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV. LV followed by _<type> indicates that an LV of the given type is required. (raid represents raid<N> type)

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Extend the size of an LV by 54MiB, using a specific PV.

lvextend -L +54 vg01/lvol10 /dev/sdk3

Extend the size of an LV by the amount of free space on PV /dev/sdk3. This is equivalent to specifying "-l +100%PVS" on the command line.

lvextend vg01/lvol01 /dev/sdk3

Extend an LV by 16MiB using specific physical extents.

lvextend -L+16m vg01/lvol01 /dev/sda:8-9 /dev/sdb:8-9

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvmconfig – Display and manipulate configuration information

SYNOPSIS

```
lvmconfig
[ option_args ]
[ position_args ]
```

DESCRIPTION

lvmconfig, **lvm config**, **lvm dumpconfig** (for compatibility reasons, to be phased out) produce formatted output from the LVM configuration tree. The sources of the configuration data include **lvm.conf(5)** and command line settings from **--config**.

USAGE

```
lvmconfig
[ -f|--file String ]
[ -l|--list ]
[ --atversion String ]
[ --typeconfig current|default|diff|full|list|missing|new|profilable|profilable-command|profilable-metadata ]
[ --ignoreadvanced ]
[ --ignoreunsupported ]
[ --ignorelocal ]
[ --mergedconfig ]
[ --metadataprofile String ]
[ --sinceversion String ]
[ --showdeprecated ]
[ --showunsupported ]
[ --validate ]
[ --withsummary ]
[ --withcomments ]
[ --withgeneralpreamble ]
[ --withlocalpreamble ]
[ --withspaces ]
[ --unconfigured ]
[ --withversions ]
[ COMMON_OPTIONS ]
[ String ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--atversion *String*

Specify an LVM version in x.y.z format where x is the major version, the y is the minor version and z is the patchlevel (e.g. 2.2.106). When configuration is displayed, the configuration settings recognized at this LVM version will be considered only. This can be used to display a configuration that a certain LVM version understands and which does not contain any newer settings for which LVM would issue a warning message when checking the configuration.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--file *String*

Write output to the named file.

-h|--help

Display help text.

--ignoreadvanced

Exclude advanced configuration settings from the output.

--ignorelocal

Ignore the local section. The local section should be defined in the lvmlocal.conf file, and should contain config settings specific to the local host which should not be copied to other hosts.

--ignoreunsupported

Exclude unsupported configuration settings from the output. These settings are either used for debugging and development purposes only or their support is not yet complete and they are not meant to be used in production. The **current** and **diff** types include unsupported settings in their output by default, all the other types ignore unsupported settings.

-l|--list

List config settings with summarizing comment. This is the same as using options --typeconfig list --withsummary.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--mergedconfig

When the command is run with --config and/or --commandprofile (or using LVM_COMMAND_PROFILE environment variable), --profile, or --metadataprofile, merge all the contents of the "config cascade" before displaying it. Without merging, only the configuration at the front of the cascade is displayed. See **lvm.conf(5)** for more information about config.

--metadataprofile *String*

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--showdeprecated

Include deprecated configuration settings in the output. These settings are deprecated after a certain version. If a concrete version is specified with --atversion, deprecated settings are automatically included if the specified version is lower than the version in which the settings were deprecated. The current and diff types include deprecated settings in their output by default, all the other types ignore deprecated settings.

--showunsupported

Include unsupported configuration settings in the output. These settings are either used for debugging or development purposes only, or their support is not yet complete and they are not meant to be used in production. The current and diff types include unsupported settings in their output by default, all the other types ignore unsupported settings.

--sinceversion String

Specify an LVM version in x.y.z format where x is the major version, the y is the minor version and z is the patchlevel (e.g. 2.2.106). This option is currently applicable only with --typeconfig new to display all configuration settings introduced since given version.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--typeconfig current|default|diff|full|list|missing|new|profilable|profilable-command|profilable-metadata

current prints the config settings that would be applied to an lvm command (assuming the command does not override them on the command line.) This includes: settings that have been modified in lvm config files, settings that get their default values from config files, and default settings that have been uncommented in config files. **default** prints all settings with their default values. Changes made in lvm config files are not reflected in the output. Some settings get their default values internally, and these settings are printed as comments. Other settings get their default values from config files, and these settings are not printed as comments. **diff** prints only config settings that have been modified from their default values in config files (the difference between current and default.) **full** prints every setting uncommented and set to the current value, i.e. how it would be used by an lvm command. This includes settings modified in config files, settings that usually get defaults internally, and settings that get defaults from config files. **list** prints all config names without values. **missing** prints settings that are missing from the lvm config files. A missing setting that usually gets its default from config files is printed uncommented and set to the internal default. Settings that get their default internally and are not set in config files are printed commented with the internal default. **new** prints config settings that have been added since the lvm version specified by --sinceversion. They are printed with their default values. **profilable** prints settings with their default values that can be set from a profile. **profilable-command** prints settings with their default values that can be set from a command profile. **profilable-metadata** prints settings with their default values that can be set from a metadata profile. Also see **lvm.conf(5)**.

--unconfigured

Internal option used for generating config file during build.

--validate

Validate current configuration used and exit with appropriate return code. The validation is done only for the configuration at the front of the "config cascade". To validate the whole merged configuration tree, also use --mergedconfig. The validation is done even if lvm.conf config/checks is disabled.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--withcomments

Display a full comment for each configuration node. For deprecated settings, also display comments about deprecation.

--withgeneralpreamble

Include general config file preamble.

--withlocalpreamble

Include local config file preamble.

--withspaces

Where appropriate, add more spaces in output for better readability.

--withsummary

Display a one line comment for each configuration node.

--withversions

Also display a comment containing the version of introduction for each configuration node. If the setting is deprecated, also display the version since which it is deprecated.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*String*

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

**lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)**
**lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)**
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvm — LVM2 tools

SYNOPSIS

lvm [*command|file*]

DESCRIPTION

The Logical Volume Manager (LVM) provides tools to create virtual block devices from physical devices. Virtual devices may be easier to manage than physical devices, and can have capabilities beyond what the physical devices provide themselves. A Volume Group (VG) is a collection of one or more physical devices, each called a Physical Volume (PV). A Logical Volume (LV) is a virtual block device that can be used by the system or applications. Each block of data in an LV is stored on one or more PV in the VG, according to algorithms implemented by Device Mapper (DM) in the kernel.

The **lvm** command, and other commands listed below, are the command-line tools for LVM. A separate manual page describes each command in detail.

If **lvm** is invoked with no arguments it presents a readline prompt (assuming it was compiled with readline support). LVM commands may be entered interactively at this prompt with readline facilities including history and command name and option completion. Refer to **readline(3)** for details.

If **lvm** is invoked with argv[0] set to the name of a specific LVM command (for example by using a hard or soft link) it acts as that command.

On invocation, **lvm** requires that only the standard file descriptors stdin, stdout and stderr are available. If others are found, they get closed and messages are issued warning about the leak. This warning can be suppressed by setting the environment variable **LVM_SUPPRESS_FD_WARNINGS**.

Where commands take VG or LV names as arguments, the full path name is optional. An LV called "lvol0" in a VG called "vg0" can be specified as "vg0/lvol0". Where a list of VGs is required but is left empty, a list of all VGs will be substituted. Where a list of LVs is required but a VG is given, a list of all the LVs in that VG will be substituted. So **lvdisplay vg0** will display all the LVs in "vg0". Tags can also be used – see **--addtag** below.

One advantage of using the built-in shell is that configuration information gets cached internally between commands.

A file containing a simple script with one command per line can also be given on the command line. The script can also be executed directly if the first line is #! followed by the absolute path of **lvm**.

Additional hyphens within option names are ignored. For example, **--readonly** and **--read-only** are both accepted.

BUILT-IN COMMANDS

The following commands are built into lvm without links normally being created in the filesystem for them.

config	The same as lvmconfig(8) below.
devtypes	Display the recognised built-in block device types.
dumpconfig	The same as lvmconfig(8) below.
formats	Display recognised metadata formats.
fullreport	Report information about PVs, PV segments, VGs, LVs and LV segments, all at once.
help	Display the help text.
lastlog	Display log report of last command run in LVM shell if command log reporting is enabled.
lvpoll	Complete lvmpolld operations (Internal command).
segtypes	Display recognised Logical Volume segment types.
systemid	Display any system ID currently set on this host.
tags	Display any tags defined on this host.
version	Display version information.

COMMANDS

The following commands implement the core LVM functionality.

pvchange	Change attributes of a Physical Volume.
pvck	Check Physical Volume metadata.
pvcreate	Initialize a disk or partition for use by LVM.
pvdisk	Display attributes of a Physical Volume.
pvmove	Move Physical Extents.
pvremove	Remove a Physical Volume.
pvresize	Resize a disk or partition in use by LVM2.
pvs	Report information about Physical Volumes.
pvscan	Scan all disks for Physical Volumes.
vgcfgbackup	Backup Volume Group descriptor area.
vgcfgrestore	Restore Volume Group descriptor area.
vgchange	Change attributes of a Volume Group.
vgck	Check Volume Group metadata.
vgconvert	Convert Volume Group metadata format.
vgcreate	Create a Volume Group.
vgdisplay	Display attributes of Volume Groups.
vgexport	Make volume Groups unknown to the system.
vgextend	Add Physical Volumes to a Volume Group.
vgimport	Make exported Volume Groups known to the system.
vgimportclone	Import and rename duplicated Volume Group (e.g. a hardware snapshot).
vgmerge	Merge two Volume Groups.
vgmknodes	Recreate Volume Group directory and Logical Volume special files
vgreduce	Reduce a Volume Group by removing one or more Physical Volumes.
vgremove	Remove a Volume Group.
vgrename	Rename a Volume Group.
vgs	Report information about Volume Groups.
vgscan	Scan all disks for Volume Groups.
vgsplit	Split a Volume Group into two, moving any logical volumes from one Volume Group to another by moving entire Physical Volumes.
lvchange	Change attributes of a Logical Volume.
lvconvert	Convert a Logical Volume from linear to mirror or snapshot.
lvcreate	Create a Logical Volume in an existing Volume Group.
lvdisplay	Display attributes of a Logical Volume.
lvextend	Extend the size of a Logical Volume.
lvmconfig	Display the configuration information after loading lvm.conf(5) and any other configuration files.
lvmdiskscan	Scan for all devices visible to LVM2.
lvmdump	Create lvm2 information dumps for diagnostic purposes.
lvreduce	Reduce the size of a Logical Volume.
lvremove	Remove a Logical Volume.
lvrename	Rename a Logical Volume.
lvresize	Resize a Logical Volume.
lvs	Report information about Logical Volumes.
lvscan	Scan (all disks) for Logical Volumes.

The following LVM1 commands are not implemented in LVM2: **lvmchange**, **lvmsadc**, **lvmsar**, **pvdata**. For performance metrics, use **dmstats(8)** or to manipulate the kernel device-mapper driver used by LVM2 directly, use **dmsetup(8)**.

VALID NAMES

The valid characters for VG and LV names are: **a–z A–Z 0–9 + _ . –**

VG names cannot begin with a hyphen. The name of a new LV also cannot begin with a hyphen. However,

if the configuration setting **metadata/record_lvs_history** is enabled then an LV name with a hyphen as a prefix indicates that, although the LV was removed, it is still being tracked because it forms part of the history of at least one LV that is still present. This helps to record the ancestry of thin snapshots even after some links in the chain have been removed. A reference to the historical LV 'lvol1' in VG 'vg00' would be 'vg00\-\lvol1' or just '\lvol1' if the VG is already set. (The latter form must be preceded by '--' to terminate command line option processing before reaching this argument.)

There are also various reserved names that are used internally by lvm that can not be used as LV or VG names. A VG cannot be called anything that exists in */dev/* at the time of creation, nor can it be called '.' or '..'. An LV cannot be called '.', '..', 'snapshot' or 'pvmove'. The LV name may also not contain any of the following strings: '_cdata', '_cmeta', '_corig', '_mlog', '_mimage', '_pmsspare', '_rimage', '_rmeta', '_tdata', '_tmeta', '_vorigin' or '_vdata'. A directory bearing the name of each Volume Group is created under */dev* when any of its Logical Volumes are activated. Each active Logical Volume is accessible from this directory as a symbolic link leading to a device node. Links or nodes in */dev/mapper* are intended only for internal use and the precise format and escaping might change between releases and distributions. Other software and scripts should use the */dev/VolumeGroupName/LogicalVolumeName* format to reduce the chance of needing amendment when the software is updated. Should you need to process the node names in */dev/mapper*, you may use **dmsetup splitname** to separate out the original VG, LV and internal layer names.

UNIQUE NAMES

VG names should be unique. **vgcreate** will produce an error if the specified VG name matches an existing VG name. However, there are cases where different VGs with the same name can appear to LVM, e.g. after moving disks or changing filters.

When VGs with the same name exist, commands operating on all VGs will include all of the VGs with the same name. If the ambiguous VG name is specified on the command line, the command will produce an error. The error states that multiple VGs exist with the specified name. To process one of the VGs specifically, the --select option should be used with the UUID of the intended VG: '--select vg_uuid=<uuid>'.

An exception is if all but one of the VGs with the shared name is foreign (see **lvmsystemid**(7).) In this case, the one VG that is not foreign is assumed to be the intended VG and is processed.

LV names are unique within a VG. The name of an historical LV cannot be reused until the historical LV has itself been removed or renamed.

ALLOCATION

When an operation needs to allocate Physical Extents for one or more Logical Volumes, the tools proceed as follows:

First of all, they generate the complete set of unallocated Physical Extents in the Volume Group. If any ranges of Physical Extents are supplied at the end of the command line, only unallocated Physical Extents within those ranges on the specified Physical Volumes are considered.

Then they try each allocation policy in turn, starting with the strictest policy (**contiguous**) and ending with the allocation policy specified using **--alloc** or set as the default for the particular Logical Volume or Volume Group concerned. For each policy, working from the lowest-numbered Logical Extent of the empty Logical Volume space that needs to be filled, they allocate as much space as possible according to the restrictions imposed by the policy. If more space is needed, they move on to the next policy.

The restrictions are as follows:

Contiguous requires that the physical location of any Logical Extent that is not the first Logical Extent of a Logical Volume is adjacent to the physical location of the Logical Extent immediately preceding it.

Cling requires that the Physical Volume used for any Logical Extent to be added to an existing Logical

Volume is already in use by at least one Logical Extent earlier in that Logical Volume. If the configuration parameter **allocation/cling_tag_list** is defined, then two Physical Volumes are considered to match if any of the listed tags is present on both Physical Volumes. This allows groups of Physical Volumes with similar properties (such as their physical location) to be tagged and treated as equivalent for allocation purposes.

When a Logical Volume is striped or mirrored, the above restrictions are applied independently to each stripe or mirror image (leg) that needs space.

Normal will not choose a Physical Extent that shares the same Physical Volume as a Logical Extent already allocated to a parallel Logical Volume (i.e. a different stripe or mirror image/leg) at the same offset within that parallel Logical Volume.

When allocating a mirror log at the same time as Logical Volumes to hold the mirror data, Normal will first try to select different Physical Volumes for the log and the data. If that's not possible and the **allocation/mirror_logs_require_separate_pvs** configuration parameter is set to 0, it will then allow the log to share Physical Volume(s) with part of the data.

When allocating thin pool metadata, similar considerations to those of a mirror log in the last paragraph apply based on the value of the **allocation/thin_pool_metadata_require_separate_pvs** configuration parameter.

If you rely upon any layout behaviour beyond that documented here, be aware that it might change in future versions of the code.

For example, if you supply on the command line two empty Physical Volumes that have an identical number of free Physical Extents available for allocation, the current code considers using each of them in the order they are listed, but there is no guarantee that future releases will maintain that property. If it is important to obtain a specific layout for a particular Logical Volume, then you should build it up through a sequence of **lvmcreate(8)** and **lvmconvert(8)** steps such that the restrictions described above applied to each step leave the tools no discretion over the layout.

To view the way the allocation process currently works in any specific case, read the debug logging output, for example by adding **-vvvv** to a command.

LOGICAL VOLUME TYPES

Some logical volume types are simple to create and can be done with a single **lvmcreate(8)** command. The linear and striped logical volume types are an example of this. Other logical volume types may require more than one command to create. The cache (**lvmcache(7)**) and thin provisioning (**lvmthin(7)**) types are examples of this.

DIAGNOSTICS

All tools return a status code of zero on success or non-zero on failure. The non-zero codes distinguish only between the broad categories of unrecognised commands, problems processing the command line arguments and any other failures. As LVM remains under active development, the code used in a specific case occasionally changes between releases. Message text may also change.

ENVIRONMENT VARIABLES

HOME

Directory containing *.lvm_history* if the internal readline shell is invoked.

LVM_OUT_FD

File descriptor to use for common output from LVM commands.

LVM_ERR_FD

File descriptor to use for error output from LVM commands.

LVM_REPORT_FD

File descriptor to use for report output from LVM commands.

LVM_COMMAND_PROFILE

Name of default command profile to use for LVM commands. This profile is overriden by direct use of **--commandprofile** command line option.

LVM_RUN_BY_DMEVENTD

This variable is normally set by dmeventd plugin to inform lvm2 command it is running from dmeventd plugin so lvm2 takes some extra action to avoid communication and deadlocks with dmeventd.

LVM_SYSTEM_DIR

Directory containing **lvm.conf(5)** and other LVM system files. Defaults to "/etc/lvm".

LVM_SUPPRESS_FD_WARNINGS

Suppress warnings about unexpected file descriptors passed into LVM.

LVM_SUPPRESS_SYSLOG

Suppress contacting syslog.

LVM_VG_NAME

The Volume Group name that is assumed for any reference to a Logical Volume that doesn't specify a path. Not set by default.

LVM_LVMPOLLD_PIDFILE

Path to the file that stores the lvmpolld process ID.

LVM_LVMPOLLD_SOCKET

Path to the socket used to communicate with lvmpolld..

LVM_LOG_FILE_EPOCH

A string of up to 32 letters appended to the log filename and followed by the process ID and a startup timestamp using this format string "%s_%d_%llu". When set, each process logs to a separate file.

LVM_LOG_FILE_MAX_LINES

If more than this number of lines are sent to the log file, the command gets aborted. Automated tests use this to terminate looping commands.

LVM_EXPECTED_EXIT_STATUS

The status anticipated when the process exits. Use ">N" to match any status greater than N. If the actual exit status matches and a log file got produced, it is deleted. **LVM_LOG_FILE_EPOCH** and **LVM_EXPECTED_EXIT_STATUS** together allow automated test scripts to discard uninteresting log data.

LVM_SUPPRESS_LOCKING_FAILURE_MESSAGES

Used to suppress warning messages when the configured locking is known to be unavailable.

DM_ABORT_ON_INTERNAL_ERRORS

Abort processing if the code detects a non-fatal internal error.

DM_DISABLE_UDEV

Avoid interaction with udev. LVM will manage the relevant nodes in /dev directly.

DM_DEBUG_WITH_LINE_NUMBERS

Prepends source file name and code line number with libdm debugging.

FILES

/etc/lvm/lvm.conf
\$HOME/.lvm_history

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)
vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8)
vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8)
vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2–activation–generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

dmsetup(8), dmstats(8), readline(3)

NAME

lvm.conf — Configuration file for LVM2

SYNOPSIS

/etc/lvm/lvm.conf

DESCRIPTION

lvm.conf is loaded during the initialisation phase of **lvm(8)**. This file can in turn lead to other files being loaded – settings read in later override earlier settings. File timestamps are checked between commands and if any have changed, all the files are reloaded.

For a description of each lvm.conf setting, run:

lvmconfig --typeconfig default --withcomments --withspaces

The settings defined in lvm.conf can be overridden by any of these extended configuration methods:

direct config override on command line

The **--config ConfigurationString** command line option takes the ConfigurationString as direct string representation of the configuration to override the existing configuration. The ConfigurationString is of exactly the same format as used in any LVM configuration file.

profile config

A profile is a set of selected customizable configuration settings that are aimed to achieve a certain characteristics in various environments or uses. It's used to override existing configuration. Normally, the name of the profile should reflect that environment or use.

There are two groups of profiles recognised: **command profiles** and **metadata profiles**.

The **command profile** is used to override selected configuration settings at global LVM command level – it is applied at the very beginning of LVM command execution and it is used throughout the whole time of LVM command execution. The command profile is applied by using the **--commandprofile ProfileName** command line option that is recognised by all LVM2 commands.

The **metadata profile** is used to override selected configuration settings at Volume Group/Logical Volume level – it is applied independently for each Volume Group/Logical Volume that is being processed. As such, each Volume Group/Logical Volume can store the profile name used in its metadata so next time the Volume Group/Logical Volume is processed, the profile is applied automatically. If Volume Group and any of its Logical Volumes have different profiles defined, the profile defined for the Logical Volume is preferred. The metadata profile can be attached/detached by using the **lvchange** and **vgchange** commands and their **--metadataprofile ProfileName** and **--detachprofile** options or the **--metadataprofile** option during creation when using **vgcreate** or **lvcreate** command. The **vgs** and **lvs** reporting commands provide **-o vg_profile** and **-o lv_profile** output options to show the metadata profile currently attached to a Volume Group or a Logical Volume.

The set of options allowed for command profiles is mutually exclusive when compared to the set of options allowed for metadata profiles. The settings that belong to either of these two sets can't be mixed together and LVM tools will reject such profiles.

LVM itself provides a few predefined configuration profiles. Users are allowed to add more profiles with different values if needed. For this purpose, there's the **command_profile_template.profile** (for command profiles) and **metadata_profile_template.profile** (for metadata profiles) which contain all settings that are customizable by profiles of certain type. Users are encouraged to copy these template profiles and edit them as needed. Alternatively, the **lvmconfig --file <ProfileName.profile> --type profitable-command <section>** or **lvmconfig --file**

<ProfileName.profile> --type profilable-metadata <section> can be used to generate a configuration with profilable settings in either of the type for given section and save it to new Profile-Name.profile (if the section is not specified, all profilable settings are reported).

The profiles are stored in /etc/lvm/profile directory by default. This location can be changed by using the **config/profile_dir** setting. Each profile configuration is stored in **ProfileName.profile** file in the profile directory. When referencing the profile, the **.profile** suffix is left out.

tag config

See **tags** configuration setting description below.

When several configuration methods are used at the same time and when LVM looks for the value of a particular setting, it traverses this **config cascade** from left to right:

direct config override on command line → **command profile config** → **metadata profile config** → **tag config** → **lvmlocal.conf** → **lvm.conf**

No part of this cascade is compulsory. If there's no setting value found at the end of the cascade, a default value is used for that setting. Use lvmconfig to check what settings are in use and what the default values are.

SYNTAX

This section describes the configuration file syntax.

Whitespace is not significant unless it is within quotes. This provides a wide choice of acceptable indentation styles. Comments begin with # and continue to the end of the line. They are treated as whitespace.

Here is an informal grammar:

file = value*

A configuration file consists of a set of values.

value = section | assignment

A value can either be a new section, or an assignment.

section = identifier '{' value* '}'

A section groups associated values together. If the same section is encountered multiple times, the contents of all instances are concatenated together in the order of appearance.

It is denoted by a name and delimited by curly brackets.

e.g. backup {

...

}

assignment = identifier '=' (array | type)

An assignment associates a type with an identifier. If the identifier contains forward slashes, those are interpreted as path delimiters. The statement **section/key = value** is equivalent to **section { key = value }**. If multiple instances of the same key are encountered, only the last value is used (and a warning is issued).

e.g. level = 7

array = '[' (type ',')* type ']' | '[' ']

Inhomogeneous arrays are supported.

Elements must be separated by commas.

An empty array is acceptable.

type = integer | float | string

integer = [0-9]*

float = [0-9]* '.' [0-9]*

string = '"' .* '"'

Strings with spaces must be enclosed in double quotes, single words that start with a letter can be left unquoted.

SETTINGS

The **lvmconfig** command prints the LVM configuration settings in various ways. See the man page **lvmconfig(8)**.

Command to print a list of all possible config settings, with their default values:
lvmconfig --type default

Command to print a list of all possible config settings, with their default values, and a full description of each as a comment:

lvmconfig --type default --withcomments

Command to print a list of all possible config settings, with their current values (configured, non-default values are shown):

lvmconfig --type current

Command to print all config settings that have been configured with a different value than the default (configured, non-default values are shown):

lvmconfig --type diff

Command to print a single config setting, with its default value, and a full description, where "Section" refers to the config section, e.g. global, and "Setting" refers to the name of the specific setting, e.g. umask:

lvmconfig --type default --withcomments Section/Setting

FILES

*/etc/lvm/lvm.conf
/etc/lvm/lvmlocal.conf
/etc/lvm/archive
/etc/lvm/backup
/etc/lvm/cache/.cache
/run/lock/lvm
/etc/lvm/profile*

SEE ALSO

lvm(8) lvmconfig(8)

lvm_selinux(8) - Linux man page

Name

`lvm_selinux` - Security Enhanced Linux Policy for the lvm processes

Description

Security-Enhanced Linux secures the lvm processes via flexible mandatory access control.

The lvm processes execute with the `lvm_t` SELinux type. You can check if you have these processes running by executing the `ps` command with the `-Z` qualifier.

For example:

```
ps -eZ | grep lvm_t
```

Entrypoints

The `lvm_t` SELinux type can be entered via the
"mtrr_device_t,unlabeled_t,proc_type,sysctl_type,filesystem_type,file_type,lvm_exec_t"
file types. The default entrypoint paths for the `lvm_t` domain are the following:"

```
/dev/cpu/mtrr, all files on the system, /lib/lvm-10/.*, /lib/lvm-200/.*, /sbin/lvs, /sbin/vgs,  
/sbin/pvs, /sbin/lvm, /sbin/vgck, /sbin/pvdata, /sbin/pvmove, /sbin/pvscan, /sbin/lvscan,  
/sbin/kpartx, /sbin/lvmsar, /sbin/dmraid, /sbin/vgscan, /sbin/vgmerge, /sbin/dmsetup,  
/sbin/e2fsadm, /sbin/lvmsadc, /sbin/lvmetad, /sbin/vgsplit, /usr/sbin/lvm, /sbin/vgchange,  
/sbin/vgexport, /sbin/vgcreate, /sbin/vgextend, /sbin/vgimport, /sbin/vgreduce,  
/sbin/vgremove, /sbin/vgrename, /sbin/pvchange, /sbin/pvcreate, /sbin/pvremove,  
/sbin/lvreduce, /sbin/lvrename, /sbin/lvresize, /sbin/lvremove, /sbin/lvchange,  
/sbin/lvextend, /sbin/lvcreate, /sbin/vgdisplay, /sbin/vgmknodes, /sbin/pvdisplay,  
/sbin/lvdisplay, /sbin/lvmchange, /sbin/vgwrapper, /sbin/multipathd, /sbin/lvm.static,  
/sbin/cryptsetup, /sbin/vgcfgbackup, /sbin/lvmdiskscan, /sbin/mount.crypt,  
/sbin/vgcfgrestore, /sbin/lvmiopversion, /sbin/vgscan.static, /sbin/dmsetup.static,  
/sbin/vgchange.static, /sbin/multipath.static, /lib/udev/udisks-lvm-pv-export
```

Process Types

SELinux defines process types (domains) for each process running on the system

You can see the context of a process using the `-Z` option to `ps`

Policy governs the access confined processes have to files. SELinux lvm policy is very flexible allowing users to setup their lvm processes in as secure a method as possible.

The following process types are defined for lvm:

lvm_t

Note: **semanage permissive -a lvm_t**

can be used to make the process type lvm_t permissive. Permissive process types are not denied access by SELinux. AVC messages will still be generated.

File Contexts

SELinux requires files to have an extended attribute to define the file type.

You can see the context of a file using the **-Z** option to **ls**

Policy governs the access confined processes have to these files. SELinux lvm policy is very flexible allowing users to setup their lvm processes in as secure a method as possible.

The following file types are defined for lvm:

lvm_etc_t

- Set files with the lvm_etc_t type, if you want to store lvm files in the /etc directories.

lvm_exec_t

- Set files with the lvm_exec_t type, if you want to transition an executable to the lvm_t domain.

lvm_lock_t

- Set files with the lvm_lock_t type, if you want to treat the files as lvm lock data, stored under the /var/lock directory

lvm_metadata_t

- Set files with the lvm_metadata_t type, if you want to treat the files as lvm metadata data.

lvm_tmp_t

- Set files with the lvm_tmp_t type, if you want to store lvm temporary files in the /tmp directories.

lvm_var_lib_t

- Set files with the lvm_var_lib_t type, if you want to store the lvm files under the /var/lib directory.

lvm_var_run_t

- Set files with the lvm_var_run_t type, if you want to store the lvm files under the /run directory.

Note: File context can be temporarily modified with the chcon command. If you want to permanently change the file context you need to use the **semanage fcontext** command. This will modify the SELinux labeling database. You will need to use **restorecon** to apply the labels.

Managed Files

The SELinux process type lvm_t can manage files labeled with the following file types. The paths listed are the default paths for these file types. Note the processes UID still need to have DAC permissions.

file_type

all files on the system

Commands

semanage fcontext can also be used to manipulate default file context mappings.

semanage permissive can also be used to manipulate whether or not a process type is permissive.

semanage module can also be used to enable/disable/install/remove policy modules.

system-config-selinux is a GUI tool available to customize SELinux policy settings.

Author

This manual page was auto-generated using **sepolicy manpage** by mgrepl.

See Also

[**selinux**\(8\)](#), [**lvm**\(8\)](#), [**semanage**\(8\)](#), [**restorecon**\(8\)](#), [**chcon**\(1\)](#), [**sepolicy**\(8\)](#)

NAME

lvmconfig – Display and manipulate configuration information

SYNOPSIS

```
lvmconfig
[ option_args ]
[ position_args ]
```

DESCRIPTION

lvmconfig, **lvm config**, **lvm dumpconfig** (for compatibility reasons, to be phased out) produce formatted output from the LVM configuration tree. The sources of the configuration data include **lvm.conf(5)** and command line settings from **--config**.

USAGE

```
lvmconfig
[ -f|--file String ]
[ -l|--list ]
[ --atversion String ]
[ --typeconfig current|default|diff|full|list|missing|new|profilable|profilable-command|profilable-metadata ]
[ --ignoreadvanced ]
[ --ignoreunsupported ]
[ --ignorelocal ]
[ --mergedconfig ]
[ --metadataprofile String ]
[ --sinceversion String ]
[ --showdeprecated ]
[ --showunsupported ]
[ --validate ]
[ --withsummary ]
[ --withcomments ]
[ --withgeneralpreamble ]
[ --withlocalpreamble ]
[ --withspaces ]
[ --unconfigured ]
[ --withversions ]
[ COMMON_OPTIONS ]
[ String ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--atversion *String*

Specify an LVM version in x.y.z format where x is the major version, the y is the minor version and z is the patchlevel (e.g. 2.2.106). When configuration is displayed, the configuration settings recognized at this LVM version will be considered only. This can be used to display a configuration that a certain LVM version understands and which does not contain any newer settings for which LVM would issue a warning message when checking the configuration.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--file *String*

Write output to the named file.

-h|--help

Display help text.

--ignoreadvanced

Exclude advanced configuration settings from the output.

--ignorelocal

Ignore the local section. The local section should be defined in the lvmlocal.conf file, and should contain config settings specific to the local host which should not be copied to other hosts.

--ignoreunsupported

Exclude unsupported configuration settings from the output. These settings are either used for debugging and development purposes only or their support is not yet complete and they are not meant to be used in production. The **current** and **diff** types include unsupported settings in their output by default, all the other types ignore unsupported settings.

-l|--list

List config settings with summarizing comment. This is the same as using options --typeconfig list --withsummary.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--mergedconfig

When the command is run with --config and/or --commandprofile (or using LVM_COMMAND_PROFILE environment variable), --profile, or --metadataprofile, merge all the contents of the "config cascade" before displaying it. Without merging, only the configuration at the front of the cascade is displayed. See **lvm.conf(5)** for more information about config.

--metadataprofile *String*

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--showdeprecated

Include deprecated configuration settings in the output. These settings are deprecated after a certain version. If a concrete version is specified with --atversion, deprecated settings are automatically included if the specified version is lower than the version in which the settings were deprecated. The current and diff types include deprecated settings in their output by default, all the other types ignore deprecated settings.

--showunsupported

Include unsupported configuration settings in the output. These settings are either used for debugging or development purposes only, or their support is not yet complete and they are not meant to be used in production. The current and diff types include unsupported settings in their output by default, all the other types ignore unsupported settings.

--sinceversion String

Specify an LVM version in x.y.z format where x is the major version, the y is the minor version and z is the patchlevel (e.g. 2.2.106). This option is currently applicable only with --typeconfig new to display all configuration settings introduced since given version.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--typeconfig current|default|diff|full|list|missing|new|profilable|profilable-command|profilable-metadata

current prints the config settings that would be applied to an lvm command (assuming the command does not override them on the command line.) This includes: settings that have been modified in lvm config files, settings that get their default values from config files, and default settings that have been uncommented in config files. **default** prints all settings with their default values. Changes made in lvm config files are not reflected in the output. Some settings get their default values internally, and these settings are printed as comments. Other settings get their default values from config files, and these settings are not printed as comments. **diff** prints only config settings that have been modified from their default values in config files (the difference between current and default.) **full** prints every setting uncommented and set to the current value, i.e. how it would be used by an lvm command. This includes settings modified in config files, settings that usually get defaults internally, and settings that get defaults from config files. **list** prints all config names without values. **missing** prints settings that are missing from the lvm config files. A missing setting that usually gets its default from config files is printed uncommented and set to the internal default. Settings that get their default internally and are not set in config files are printed commented with the internal default. **new** prints config settings that have been added since the lvm version specified by --sinceversion. They are printed with their default values. **profilable** prints settings with their default values that can be set from a profile. **profilable-command** prints settings with their default values that can be set from a command profile. **profilable-metadata** prints settings with their default values that can be set from a metadata profile. Also see **lvm.conf(5)**.

--unconfigured

Internal option used for generating config file during build.

--validate

Validate current configuration used and exit with appropriate return code. The validation is done only for the configuration at the front of the "config cascade". To validate the whole merged configuration tree, also use --mergedconfig. The validation is done even if lvm.conf config/checks is disabled.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--withcomments

Display a full comment for each configuration node. For deprecated settings, also display comments about deprecation.

--withgeneralpreamble

Include general config file preamble.

--withlocalpreamble

Include local config file preamble.

--withspaces

Where appropriate, add more spaces in output for better readability.

--withsummary

Display a one line comment for each configuration node.

--withversions

Also display a comment containing the version of introduction for each configuration node. If the setting is deprecated, also display the version since which it is deprecated.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*String*

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

**lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)**
**lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)**
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

`lvmraid` — LVM RAID

DESCRIPTION

lvm(8) RAID is a way to create a Logical Volume (LV) that uses multiple physical devices to improve performance or tolerate device failures. In LVM, the physical devices are Physical Volumes (PVs) in a single Volume Group (VG).

How LV data blocks are placed onto PVs is determined by the RAID level. RAID levels are commonly referred to as 'raid' followed by a number, e.g. raid1, raid5 or raid6. Selecting a RAID level involves making tradeoffs among: physical device requirements, fault tolerance, and performance. A description of the RAID levels can be found at

www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf

LVM RAID uses both Device Mapper (DM) and Multiple Device (MD) drivers from the Linux kernel. DM is used to create and manage visible LVM devices, and MD is used to place data on physical devices.

LVM creates hidden LVs (dm devices) layered between the visible LV and physical devices. LVs in the middle layers are called sub LVs. For LVM raid, a sub LV pair to store data and metadata (raid superblock and write intent bitmap) is created per raid image/leg (see `lvs` command examples below).

Create a RAID LV

To create a RAID LV, use `lvcreate` and specify an LV type. The LV type corresponds to a RAID level. The basic RAID levels that can be used are: **raid0, raid1, raid4, raid5, raid6, raid10**.

lvcreate --type RaidLevel [OPTIONS] --name Name --size Size VG [PVs]

To display the LV type of an existing LV, run:

lvs -o name,segtype LV

(The LV type is also referred to as "segment type" or "segtype".)

LVs can be created with the following types:

raid0

Also called striping, raid0 spreads LV data across multiple devices in units of stripe size. This is used to increase performance. LV data will be lost if any of the devices fail.

lvcreate --type raid0 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to spread the LV across.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose *Number* devices, one for each stripe based on the number of PVs available or supplied.

raid1

Also called mirroring, raid1 uses multiple devices to duplicate LV data. The LV data remains available if all but one of the devices fail. The minimum number of devices (i.e. sub LV pairs) required is 2.

lvmcreate --type raid1 [--mirrors Number] VG [PVs]

--mirrors specifies the number of mirror images in addition to the original LV image, e.g. **--mirrors 1** means there are two images of the data, the original and one mirror image.

PVs specifies the devices to use. If not specified, lvm will choose *Number* devices, one for each image.

raid4

raid4 is a form of striping that uses an extra, first device dedicated to storing parity blocks. The LV data remains available if one device fails. The parity is used to recalculate data that is lost from a single device. The minimum number of devices required is 3.

lvmcreate --type raid4 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra device lvm adds for storing parity blocks. A raid4 LV with *Number* stripes requires *Number+1* devices. *Number* must be 2 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose *Number+1* separate devices.

raid4 is called non-rotating parity because the parity blocks are always stored on the same device.

raid5

raid5 is a form of striping that uses an extra device for storing parity blocks. LV data and parity blocks are stored on each device, typically in a rotating pattern for performance reasons. The LV data remains available if one device fails. The parity is used to recalculate data that is lost from a single device. The minimum number of devices required is 3 (unless converting from 2 legged raid1 to reshape to more stripes; see reshaping).

lvmcreate --type raid5 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra device lvm adds for storing parity blocks. A raid5 LV with *Number* stripes requires *Number+1* devices. *Number* must be 2 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose *Number+1* separate devices.

raid5 is called rotating parity because the parity blocks are placed on different devices in a round-robin sequence. There are variations of raid5 with different algorithms for placing the parity blocks. The default variant is raid5_ls (raid5 left symmetric, which is a rotating parity 0 with data restart.) See **RAID5 variants** below.

raid6

raid6 is a form of striping like raid5, but uses two extra devices for parity blocks. LV data and parity blocks are stored on each device, typically in a rotating pattern for performance reasons. The LV data remains available if up to two devices fail. The parity is used to recalculate data that is lost from one or two devices. The minimum number of devices required is 5.

lvmcreate --type raid6 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra two devices lvm adds for storing parity blocks. A raid6 LV with *Number* stripes requires *Number+2* devices. *Number* must be 3 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose *Number+2* separate devices.

Like raid5, there are variations of raid6 with different algorithms for placing the parity blocks. The default variant is raid6_zr (raid6 zero restart, aka left symmetric, which is a rotating parity 0 with data restart.) See **RAID6 variants** below.

raid10

raid10 is a combination of raid1 and raid0, striping data across mirrored devices. LV data remains available if one or more devices remain in each mirror set. The minimum number of devices required is 4.

lvmcreate --type raid10
[--mirrors NumberMirrors]
[--stripes NumberStripes --stripesize Size]
VG [PVs]

--mirrors specifies the number of mirror images within each stripe. e.g. --mirrors 1 means there are two images of the data, the original and one mirror image.

--stripes specifies the total number of devices to use in all raid1 images (not the number of raid1 devices to spread the LV across, even though that is the effective result). The number of devices in each raid1 mirror will be *NumberStripes/NumberMirrors+1*, e.g. mirrors 1 and stripes 4 will stripe data across two raid1 mirrors, where each mirror is 2 devices.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose the necessary devices. Devices are used to create mirrors in the order listed, e.g. for mirrors 1, stripes 2, listing PV1 PV2 PV3 PV4 results in mirrors PV1/PV2 and PV3/PV4.

RAID10 is not mirroring on top of stripes, which would be RAID01, which is less tolerant of device failures.

Synchronization

Synchronization is the process that makes all the devices in a RAID LV consistent with each other.

In a RAID1 LV, all mirror images should have the same data. When a new mirror image is added, or a mirror image is missing data, then images need to be synchronized. Data blocks are copied from an existing image to a new or outdated image to make them match.

In a RAID 4/5/6 LV, parity blocks and data blocks should match based on the parity calculation. When the devices in a RAID LV change, the data and parity blocks can become inconsistent and need to be synchronized. Correct blocks are read, parity is calculated, and recalculated blocks are written.

The RAID implementation keeps track of which parts of a RAID LV are synchronized. When a RAID LV is first created and activated the first synchronization is called initialization. A pointer stored in the raid metadata keeps track of the initialization process thus allowing it to be restarted after a deactivation of the RaidLV or a crash. Any writes to the RaidLV dirties the respective region of the write intent bitmap which allow for fast recovery of the regions after a crash. Without this, the entire LV would need to be synchronized every time it was activated.

Automatic synchronization happens when a RAID LV is activated, but it is usually partial because the bitmaps reduce the areas that are checked. A full sync becomes necessary when devices in the RAID LV are replaced.

The synchronization status of a RAID LV is reported by the following command, where "Cpy%Sync" = "100%" means sync is complete:

```
lvs -a -o name,sync_percent
```

Scrubbing

Scrubbing is a full scan of the RAID LV requested by a user. Scrubbing can find problems that are missed by partial synchronization.

Scrubbing assumes that RAID metadata and bitmaps may be inaccurate, so it verifies all RAID metadata, LV data, and parity blocks. Scrubbing can find inconsistencies caused by hardware errors or degradation. These kinds of problems may be undetected by automatic synchronization which excludes areas outside of the RAID write-intent bitmap.

The command to scrub a RAID LV can operate in two different modes:

```
lvchange --syncaction check|repair LV
```

check Check mode is read-only and only detects inconsistent areas in the RAID LV, it does not correct them.

repair Repair mode checks and writes corrected blocks to synchronize any inconsistent areas.

Scrubbing can consume a lot of bandwidth and slow down application I/O on the RAID LV. To control the I/O rate used for scrubbing, use:

--maxrecoveryrate *Size[k|UNIT]*

Sets the maximum recovery rate for a RAID LV. *Size* is specified as an amount per second for each device in the array. If no suffix is given, then KiB/sec/device is used. Setting the recovery rate to **0** means it will be unbounded.

--minrecoveryrate *Size[k|UNIT]*

Sets the minimum recovery rate for a RAID LV. *Size* is specified as an amount per second for each device in the array. If no suffix is given, then KiB/sec/device is used. Setting the recovery rate to **0** means it will be unbounded.

To display the current scrubbing in progress on an LV, including the syncaction mode and percent complete, run:

```
lvs -a -o name,raid_sync_action,sync_percent
```

After scrubbing is complete, to display the number of inconsistent blocks found, run:

```
lvs -o name,raid_mismatch_count
```

Also, if mismatches were found, the lvs attr field will display the letter "m" (mismatch) in the 9th position, e.g.

```
# lvs -o name,vgname,segtype,attr vg/lv
LV VG Type Attr
lv vg raid1 Rwi-a-r-m-
```

Scrubbing Limitations

The **check** mode can only report the number of inconsistent blocks, it cannot report which blocks are inconsistent. This makes it impossible to know which device has errors, or if the errors affect file system data, metadata or nothing at all.

The **repair** mode can make the RAID LV data consistent, but it does not know which data is correct. The result may be consistent but incorrect data. When two different blocks of data must be made consistent, it chooses the block from the device that would be used during RAID initialization. However, if the PV holding corrupt data is known, lvchange --rebuild can be used in place of scrubbing to reconstruct the data on the bad device.

Future developments might include:

Allowing a user to choose the correct version of data during repair.

Using a majority of devices to determine the correct version of data to use in a 3-way RAID1 or RAID6 LV.

Using a checksumming device to pin-point when and where an error occurs, allowing it to be rewritten.

SubLVs

An LV is often a combination of other hidden LVs called SubLVs. The SubLVs either use physical devices, or are built from other SubLVs themselves. SubLVs hold LV data blocks, RAID parity blocks, and RAID metadata. SubLVs are generally hidden, so the lvs -a option is required to display them:

lvs -a -o name,segtype,devices

SubLV names begin with the visible LV name, and have an automatic suffix indicating its role:

- SubLVs holding LV data or parity blocks have the suffix _rimage_#. These SubLVs are sometimes referred to as DataLVs.
- SubLVs holding RAID metadata have the suffix _rmeta#. RAID metadata includes superblock information, RAID type, bitmap, and device health information. These SubLVs are sometimes referred to as MetaLVs.

SubLVs are an internal implementation detail of LVM. The way they are used, constructed and named may change.

The following examples show the SubLV arrangement for each of the basic RAID LV types, using the fewest number of devices allowed for each.

Examples**raid0**

Each rimage SubLV holds a portion of LV data. No parity is used. No RAID metadata is used.

```
# lvcreate --type raid0 --stripes 2 --name lvr0 ...
```

```
# lvs -a -o name,segtype,devices
lvr0      raid0  lvr0_rimage_0(0),lvr0_rimage_1(0)
[lvr0_rimage_0] linear /dev/sda(...)
[lvr0_rimage_1] linear /dev/sdb(...)
```

raid1

Each rimage SubLV holds a complete copy of LV data. No parity is used. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid1 --mirrors 1 --name lvr1 ...
```

```
# lvs -a -o name,segtype,devices
lvr1      raid1  lvr1_rimage_0(0),lvr1_rimage_1(0)
[lvr1_rimage_0] linear /dev/sda(...)
[lvr1_rimage_1] linear /dev/sdb(...)
[lvr1_rmeta_0] linear /dev/sda(...)
[lvr1_rmeta_1] linear /dev/sdb(...)
```

raid4

At least three rimage SubLVs each hold a portion of LV data and one rimage SubLV holds parity. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid4 --stripes 2 --name lvr4 ...
```

```
# lvs -a -o name,segtype,devices
lvr4      raid4  lvr4_rimage_0(0),\
            lvr4_rimage_1(0),\
            lvr4_rimage_2(0)
[lvr4_rimage_0] linear /dev/sda(...)
[lvr4_rimage_1] linear /dev/sdb(...)
```

```
[lvr4_rimage_2] linear /dev/sdc(...)
[lvr4_rmeta_0] linear /dev/sda(...)
[lvr4_rmeta_1] linear /dev/sdb(...)
[lvr4_rmeta_2] linear /dev/sdc(...)
```

raid5

At least three rimage SubLVs each typically hold a portion of LV data and parity (see section on raid5)
Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid5 --stripes 2 --name lvr5 ...
```

```
# lvs -a -o name,segtype,devices
lvr5      raid5  lvr5_rimage_0(0),\
            lvr5_rimage_1(0),\
            lvr5_rimage_2(0)
[lvr5_rimage_0] linear /dev/sda(...)
[lvr5_rimage_1] linear /dev/sdb(...)
[lvr5_rimage_2] linear /dev/sdc(...)
[lvr5_rmeta_0] linear /dev/sda(...)
[lvr5_rmeta_1] linear /dev/sdb(...)
[lvr5_rmeta_2] linear /dev/sdc(...)
```

raid6

At least five rimage SubLVs each typically hold a portion of LV data and parity. (see section on raid6)
Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid6 --stripes 3 --name lvr6
```

```
# lvs -a -o name,segtype,devices
lvr6      raid6  lvr6_rimage_0(0),\
            lvr6_rimage_1(0),\
            lvr6_rimage_2(0),\
            lvr6_rimage_3(0),\
            lvr6_rimage_4(0),\
            lvr6_rimage_5(0)
[lvr6_rimage_0] linear /dev/sda(...)
[lvr6_rimage_1] linear /dev/sdb(...)
[lvr6_rimage_2] linear /dev/sdc(...)
[lvr6_rimage_3] linear /dev/sdd(...)
[lvr6_rimage_4] linear /dev/sde(...)
[lvr6_rimage_5] linear /dev/sdf...
[lvr6_rmeta_0] linear /dev/sda(...)
[lvr6_rmeta_1] linear /dev/sdb(...)
[lvr6_rmeta_2] linear /dev/sdc...
[lvr6_rmeta_3] linear /dev/sdd...
[lvr6_rmeta_4] linear /dev/sde...
[lvr6_rmeta_5] linear /dev/sdf...)
```

raid10

At least four rimage SubLVs each hold a portion of LV data. No parity is used. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid10 --stripes 2 --mirrors 1 --name lvr10
```

```
# lvs -a -o name,segtype,devices
lvr10      raid10 lvr10_rimage_0(0),\
             lvr10_rimage_1(0),\
             lvr10_rimage_2(0),\
             lvr10_rimage_3(0)
[lvr10_rimage_0] linear /dev/sda(...)
[lvr10_rimage_1] linear /dev/sdb(...)
[lvr10_rimage_2] linear /dev/sdc(...)
[lvr10_rimage_3] linear /dev/sdd(...)
[lvr10_rmeta_0] linear /dev/sda(...)
[lvr10_rmeta_1] linear /dev/sdb(...)
[lvr10_rmeta_2] linear /dev/sdc(...)
[lvr10_rmeta_3] linear /dev/sdd(...)
```

Device Failure

Physical devices in a RAID LV can fail or be lost for multiple reasons. A device could be disconnected, permanently failed, or temporarily disconnected. The purpose of RAID LVs (levels 1 and higher) is to continue operating in a degraded mode, without losing LV data, even after a device fails. The number of devices that can fail without the loss of LV data depends on the RAID level:

- RAID0 (striped) LVs cannot tolerate losing any devices. LV data will be lost if any devices fail.
- RAID1 LVs can tolerate losing all but one device without LV data loss.
- RAID4 and RAID5 LVs can tolerate losing one device without LV data loss.
- RAID6 LVs can tolerate losing two devices without LV data loss.
- RAID10 is variable, and depends on which devices are lost. It stripes across multiple mirror groups with raid1 layout thus it can tolerate losing all but one device in each of these groups without LV data loss.

If a RAID LV is missing devices, or has other device-related problems, lvs reports this in the health_status (and attr) fields:

lvs -o name,lv_health_status

partial

Devices are missing from the LV. This is also indicated by the letter "p" (partial) in the 9th position of the lvs attr field.

refresh needed

A device was temporarily missing but has returned. The LV needs to be refreshed to use the device again (which will usually require partial synchronization). This is also indicated by the letter "r" (refresh needed) in the 9th position of the lvs attr field. See **Refreshing an LV**. This could also indicate a problem with the device, in which case it should be replaced, see **Replacing Devices**.

mismatches exist

See **Scrubbing**.

Most commands will also print a warning if a device is missing, e.g.

WARNING: Device for PV uItL3Z-wBME-DQy0-... not found or rejected ...

This warning will go away if the device returns or is removed from the VG (see **vgreduce --removemissing**).

Activating an LV with missing devices

A RAID LV that is missing devices may be activated or not, depending on the "activation mode" used in lvchange:

lvchange -ay --activationmode complete|degraded|partial LV

complete

The LV is only activated if all devices are present.

degraded

The LV is activated with missing devices if the RAID level can tolerate the number of missing devices without LV data loss.

partial

The LV is always activated, even if portions of the LV data are missing because of the missing device(s). This should only be used to perform extreme recovery or repair operations.

lvm.conf(5) activation/activation_mode

controls the activation mode when not specified by the command.

The default value is printed by:

lvmconfig --type default activation/activation_mode

Replacing Devices

Devices in a RAID LV can be replaced by other devices in the VG. When replacing devices that are no longer visible on the system, use lvconvert --repair. When replacing devices that are still visible, use lvconvert --replace. The repair command will attempt to restore the same number of data LVs that were previously in the LV. The replace option can be repeated to replace multiple PVs. Replacement devices can be optionally listed with either option.

lvconvert --repair LV [NewPVs]

lvconvert --replace OldPV LV [NewPV]

lvconvert --replace OldPV1 --replace OldPV2 LV [NewPVs]

New devices require synchronization with existing devices, see **Synchronization**.

Refreshing an LV

Refreshing a RAID LV clears any transient device failures (device was temporarily disconnected) and returns the LV to its fully redundant mode. Restoring a device will usually require at least partial synchronization (see **Synchronization**). Failure to clear a transient failure results in the RAID LV operating in degraded mode until it is reactivated. Use the lvchange command to refresh an LV:

lvchange --refresh LV

```
# lvs -o name,vgname,segtype,attr,size vg
```

```

LV VG Type Attr   LSize
lv vg  raid1 Rwi-a-r-r- 100.00g

# lvchange --refresh vg/lv

# lvs -o name,vgname,segtype,attr,size vg
LV VG Type Attr   LSize
lv vg  raid1 Rwi-a-r--- 100.00g

```

Automatic repair

If a device in a RAID LV fails, device-mapper in the kernel notifies the **dmeventd(8)** monitoring process (see **Monitoring**). dmeventd can be configured to automatically respond using:

lvm.conf(5) activation/raid_fault_policy

Possible settings are:

warn

A warning is added to the system log indicating that a device has failed in the RAID LV. It is left to the user to repair the LV, e.g. replace failed devices.

allocate

dmeventd automatically attempts to repair the LV using spare devices in the VG. Note that even a transient failure is treated as a permanent failure under this setting. A new device is allocated and full synchronization is started.

The specific command run by dmeventd to warn or repair is:

lvconvert --repair --use-policies LV

Corrupted Data

Data on a device can be corrupted due to hardware errors without the device ever being disconnected or there being any fault in the software. This should be rare, and can be detected (see **Scrubbing**).

Rebuild specific PVs

If specific PVs in a RAID LV are known to have corrupt data, the data on those PVs can be reconstructed with:

lvchange --rebuild PV LV

The rebuild option can be repeated with different PVs to replace the data on multiple PVs.

Monitoring

When a RAID LV is activated the **dmeventd(8)** process is started to monitor the health of the LV. Various events detected in the kernel can cause a notification to be sent from device-mapper to the monitoring process, including device failures and synchronization completion (e.g. for initialization or scrubbing).

The LVM configuration file contains options that affect how the monitoring process will respond to failure events (e.g. `raid_fault_policy`). It is possible to turn on and off monitoring with `lvchange`, but it is not recommended to turn this off unless you have a thorough knowledge of the consequences.

Configuration Options

There are a number of options in the LVM configuration file that affect the behavior of RAID LVs. The tunable options are listed below. A detailed description of each can be found in the LVM configuration file itself.

```
mirror_sectype_default
raid10_sectype_default
raid_region_size
raid_fault_policy
activation_mode
```

Data Integrity

The device mapper integrity target can be used in combination with RAID levels 1,4,5,6,10 to detect and correct data corruption in RAID images. A dm-integrity layer is placed above each RAID image, and an extra sub LV is created to hold integrity metadata (data checksums) for each RAID image. When data is read from an image, integrity checksums are used to detect corruption. If detected, dm-raid reads the data from another (good) image to return to the caller. dm-raid will also automatically write the good data back to the image with bad data to correct the corruption.

When creating a RAID LV with integrity, or adding integrity, space is required for integrity metadata. Every 500MB of LV data requires an additional 4MB to be allocated for integrity metadata, for each RAID image.

Create a RAID LV with integrity:

```
lvcreate --type raidN --raidintegrity y
```

Add integrity to an existing RAID LV:

```
lvconvert --raidintegrity y LV
```

Remove integrity from a RAID LV:

```
lvconvert --raidintegrity n LV
```

Integrity options

--raidintegritymode journal|bitmap

Use a journal (default) or bitmap for keeping integrity checksums consistent in case of a crash. The bitmap areas are recalculated after a crash, so corruption in those areas would not be detected. A journal does not have this problem. The journal mode doubles writes to storage, but can improve performance for scattered writes packed into a single journal write. bitmap mode can in theory achieve full write throughput of the device, but would not benefit from the potential scattered write optimization.

--raidintegrityblocksize 512|1024|2048|4096

The block size to use for dm-integrity on raid images. The integrity block size should usually match the device logical block size, or the file system sector/block sizes. It may be less than the file system sector/block size, but not less than the device logical block size. Possible values: 512, 1024, 2048, 4096.

Integrity initialization

When integrity is added to an LV, the kernel needs to initialize the integrity metadata (checksums) for all blocks in the LV. The data corruption checking performed by dm-integrity will only operate on areas of the

LV that are already initialized. The progress of integrity initialization is reported by the "syncpercent" LV reporting field (and under the Cpy%Sync lvs column.)

Integrity limitations

To work around some limitations, it is possible to remove integrity from the LV, make the change, then add integrity again. (Integrity metadata would need to be initialized when added again.)

LVM must be able to allocate the integrity metadata sub LV on a single PV that is already in use by the associated RAID image. This can potentially cause a problem during lvextend if the original PV holding the image and integrity metadata is full. To work around this limitation, remove integrity, extend the LV, and add integrity again.

Additional RAID images can be added to raid1 LVs, but not to other raid levels.

A raid1 LV with integrity cannot be converted to linear (remove integrity to do this.)

RAID LVs with integrity cannot yet be used as sub LVs with other LV types.

The following are not yet permitted on RAID LVs with integrity: lvreduce, pvmove, snapshots, splitmirror, raid syncaction commands, raid rebuild.

RAID1 Tuning

A RAID1 LV can be tuned so that certain devices are avoided for reading while all devices are still written to.

lvchange --[raid]writemostly PV[:y|n|t] LV

The specified device will be marked as "write mostly", which means that reading from this device will be avoided, and other devices will be preferred for reading (unless no other devices are available.) This minimizes the I/O to the specified device.

If the PV name has no suffix, the write mostly attribute is set. If the PV name has the suffix :**n**, the write mostly attribute is cleared, and the suffix :**t** toggles the current setting.

The write mostly option can be repeated on the command line to change multiple devices at once.

To report the current write mostly setting, the lvs attr field will show the letter "w" in the 9th position when write mostly is set:

lvs -a -o name,attr

When a device is marked write mostly, the maximum number of outstanding writes to that device can be configured. Once the maximum is reached, further writes become synchronous. When synchronous, a write to the LV will not complete until writes to all the mirror images are complete.

lvchange --[raid]writebehind Number LV

To report the current write behind setting, run:

lvs -o name,raid_write_behind

When write behind is not configured, or set to 0, all LV writes are synchronous.

RAID Takeover

RAID takeover is converting a RAID LV from one RAID level to another, e.g. raid5 to raid6. Changing the RAID level is usually done to increase or decrease resilience to device failures or to restripe LVs. This is done using `lvconvert` and specifying the new RAID level as the LV type:

```
lvconvert --type RaidLevel LV [PVs]
```

The most common and recommended RAID takeover conversions are:

linear to raid1

Linear is a single image of LV data, and converting it to raid1 adds a mirror image which is a direct copy of the original linear image.

striped/raid0 to raid4/5/6

Adding parity devices to a striped volume results in raid4/5/6.

Unnatural conversions that are not recommended include converting between striped and non-striped types. This is because file systems often optimize I/O patterns based on device striping values. If those values change, it can decrease performance.

Converting to a higher RAID level requires allocating new SubLVs to hold RAID metadata, and new SubLVs to hold parity blocks for LV data. Converting to a lower RAID level removes the SubLVs that are no longer needed.

Conversion often requires full synchronization of the RAID LV (see [Synchronization](#)). Converting to RAID1 requires copying all LV data blocks to N new images on new devices. Converting to a parity RAID level requires reading all LV data blocks, calculating parity, and writing the new parity blocks. Synchronization can take a long time depending on the throughput of the devices used and the size of the RaidLV. It can degrade performance (rate controls also apply to conversion; see `--minrecoverystart` and `--maxrecoverystart`.)

Warning: though it is possible to create **striped** LVs with up to 128 stripes, a maximum of 64 stripes can be converted to **raid0**, 63 to **raid4/5** and 62 to **raid6** because of the added parity SubLVs. **A striped** LV with a maximum of 32 stripes can be converted to **raid10**.

The following takeover conversions are currently possible:

- between striped and raid0.
- between linear and raid1.
- between mirror and raid1.
- between raid1 with two images and raid4/5.
- between striped/raid0 and raid4.
- between striped/raid0 and raid5.
- between striped/raid0 and raid6.
- between raid4 and raid5.
- between raid4/raid5 and raid6.
- between striped/raid0 and raid10.
- between striped and raid4.

Indirect conversions

Converting from one raid level to another may require multiple steps, converting first to intermediate raid levels.

linear to raid6

To convert an LV from linear to raid6:

1. convert to raid1 with two images
2. convert to raid5 (internally raid5_ls) with two images
3. convert to raid5 with three or more stripes (reshape)
4. convert to raid6 (internally raid6_ls_6)
5. convert to raid6 (internally raid6_zr, reshape)

The commands to perform the steps above are:

1. lvconvert --type raid1 --mirrors 1 LV
2. lvconvert --type raid5 LV
3. lvconvert --stripes 3 LV
4. lvconvert --type raid6 LV
5. lvconvert --type raid6 LV

The final conversion from raid6_ls_6 to raid6_zr is done to avoid the potential write/recovery performance reduction in raid6_ls_6 because of the dedicated parity device. raid6_zr rotates data and parity blocks to avoid this.

linear to striped

To convert an LV from linear to striped:

1. convert to raid1 with two images
2. convert to raid5_n
3. convert to raid5_n with five 128k stripes (reshape)
4. convert raid5_n to striped

The commands to perform the steps above are:

1. lvconvert --type raid1 --mirrors 1 LV
2. lvconvert --type raid5_n LV
3. lvconvert --stripes 5 --stripesize 128k LV
4. lvconvert --type striped LV

The raid5_n type in step 2 is used because it has dedicated parity SubLVs at the end, and can be converted to striped directly. The stripe size is increased in step 3 to add extra space for the conversion process. This step grows the LV size by a factor of five. After conversion, this extra space can be reduced (or used to grow the file system using the LV).

Reversing these steps will convert a striped LV to linear.

raid6 to striped

To convert an LV from raid6_nr to striped:

1. convert to raid6_n_6
2. convert to striped

The commands to perform the steps above are:

1. lvconvert --type raid6_n_6 LV
2. lvconvert --type striped LV

Examples

Converting an LV from **linear** to **raid1**.

```
# lvs -a -o name,segtype,size vg
  LV  Type  LSize
  lv  linear 300.00g

# lvconvert --type raid1 --mirrors 1 vg/lv

# lvs -a -o name,segtype,size vg
  LV      Type  LSize
  lv      raid1  300.00g
  [lv_rimage_0] linear 300.00g
  [lv_rimage_1] linear 300.00g
  [lv_rmeta_0]  linear  3.00m
  [lv_rmeta_1]  linear  3.00m
```

Converting an LV from **mirror** to **raid1**.

```
# lvs -a -o name,segtype,size vg
  LV      Type  LSize
  lv      mirror 100.00g
  [lv_mimage_0] linear 100.00g
  [lv_mimage_1] linear 100.00g
  [lv_mlog]    linear  3.00m

# lvconvert --type raid1 vg/lv

# lvs -a -o name,segtype,size vg
  LV      Type  LSize
  lv      raid1  100.00g
  [lv_rimage_0] linear 100.00g
  [lv_rimage_1] linear 100.00g
  [lv_rmeta_0]  linear  3.00m
  [lv_rmeta_1]  linear  3.00m
```

Converting an LV from **linear** to **raid1** (with 3 images).

```
# lvconvert --type raid1 --mirrors 2 vg/lv
```

Converting an LV from **striped** (with 4 stripes) to **raid6_n_6**.

```
# lvcreate --stripes 4 -L64M -n lv vg

# lvconvert --type raid6 vg/lv

# lvs -a -o lv_name,segtype,sync_percent,data_copies
  LV      Type  Cpy%Sync #Cpy
  lv      raid6_n_6 100.00   3
  [lv_rimage_0] linear
  [lv_rimage_1] linear
  [lv_rimage_2] linear
  [lv_rimage_3] linear
  [lv_rimage_4] linear
```

```
[lv_rimage_5] linear
[lv_rmeta_0] linear
[lv_rmeta_1] linear
[lv_rmeta_2] linear
[lv_rmeta_3] linear
[lv_rmeta_4] linear
[lv_rmeta_5] linear
```

This convert begins by allocating MetaLVs (rmeta_#) for each of the existing stripe devices. It then creates 2 additional MetaLV/DataLV pairs (rmeta_#/rimage_#) for dedicated raid6 parity.

If rotating data/parity is required, such as with raid6_nr, it must be done by reshaping (see below).

RAID Reshaping

RAID reshaping is changing attributes of a RAID LV while keeping the same RAID level. This includes changing RAID layout, stripe size, or number of stripes.

When changing the RAID layout or stripe size, no new SubLVs (MetaLVs or DataLVs) need to be allocated, but DataLVs are extended by a small amount (typically 1 extent). The extra space allows blocks in a stripe to be updated safely, and not be corrupted in case of a crash. If a crash occurs, reshaping can just be restarted.

(If blocks in a stripe were updated in place, a crash could leave them partially updated and corrupted. Instead, an existing stripe is quiesced, read, changed in layout, and the new stripe written to free space. Once that is done, the new stripe is unquiesced and used.)

Examples

(Command output shown in examples may change.)

Converting raid6_n_6 to raid6_nr with rotating data/parity.

This conversion naturally follows a previous conversion from striped/raid0 to raid6_n_6 (shown above). It completes the transition to a more traditional RAID6.

```
# lvs -o lv_name,segtype,sync_percent,data_copies
  LV      Type    Cpy%Sync #Cpy
  lv      raid6_n_6 100.00    3
  [lv_rimage_0] linear
  [lv_rimage_1] linear
  [lv_rimage_2] linear
  [lv_rimage_3] linear
  [lv_rimage_4] linear
  [lv_rimage_5] linear
  [lv_rmeta_0] linear
  [lv_rmeta_1] linear
  [lv_rmeta_2] linear
  [lv_rmeta_3] linear
  [lv_rmeta_4] linear
  [lv_rmeta_5] linear

# lvconvert --type raid6_nr vg/lv

# lvs -a -o lv_name,segtype,sync_percent,data_copies
```

```

LV      Type   Cpy%Sync #Cpy
lv      raid6_nr 100.00    3
[lv_rimage_0] linear
[lv_rimage_0] linear
[lv_rimage_1] linear
[lv_rimage_1] linear
[lv_rimage_2] linear
[lv_rimage_2] linear
[lv_rimage_3] linear
[lv_rimage_3] linear
[lv_rimage_4] linear
[lv_rimage_5] linear
[lv_rmeta_0] linear
[lv_rmeta_1] linear
[lv_rmeta_2] linear
[lv_rmeta_3] linear
[lv_rmeta_4] linear
[lv_rmeta_5] linear

```

The DataLVs are larger (additional segment in each) which provides space for out-of-place reshaping. The result is:

```

# lvs -a -o lv_name,segtype,seg_pe_ranges,dataoffset
LV      Type   PE Ranges     DOff
lv      raid6_nr lv_rimage_0:0-32 \
          lv_rimage_1:0-32 \
          lv_rimage_2:0-32 \
          lv_rimage_3:0-32
[lv_rimage_0] linear /dev/sda:0-31  2048
[lv_rimage_0] linear /dev/sda:33-33
[lv_rimage_1] linear /dev/sdaa:0-31  2048
[lv_rimage_1] linear /dev/sdaa:33-33
[lv_rimage_2] linear /dev/sdab:1-33  2048
[lv_rimage_3] linear /dev/sdac:1-33  2048
[lv_rmeta_0] linear /dev/sda:32-32
[lv_rmeta_1] linear /dev/sdaa:32-32
[lv_rmeta_2] linear /dev/sdab:0-0
[lv_rmeta_3] linear /dev/sdac:0-0

```

All segments with PE ranges '33-33' provide the out-of-place reshape space. The dataoffset column shows that the data was moved from initial offset 0 to 2048 sectors on each component DataLV.

For performance reasons the raid6_nr RaidLV can be restriped. Convert it from 3-way striped to 5-way-striped.

```

# lvconvert --stripes 5 vg/lv
Using default stripesize 64.00 KiB.
WARNING: Adding stripes to active logical volume vg/lv will \
grow it from 99 to 165 extents!
Run "lvresize -l99 vg/lv" to shrink it or use the additional \
capacity.
Logical volume vg/lv successfully converted.

```

```
# lvs vg/lv
```

```

LV  VG   Attr   LSize  Cpy%Sync
lv  vg    rwi-a-r-s- 652.00m 52.94

# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
LV      Attr     Type   PE Ranges      DOff
lv      rwi-a-r--- raid6_nr lv_rimage_0:0-33 \
                  lv_rimage_1:0-33 \
                  lv_rimage_2:0-33 ... \
                  lv_rimage_5:0-33 \
                  lv_rimage_6:0-33  0
[lv_rimage_0] iwi-aor--- linear /dev/sda:0-32  0
[lv_rimage_0] iwi-aor--- linear /dev/sda:34-34
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:0-32  0
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:34-34
[lv_rimage_2] iwi-aor--- linear /dev/sdab:0-32  0
[lv_rimage_2] iwi-aor--- linear /dev/sdab:34-34
[lv_rimage_3] iwi-aor--- linear /dev/sdac:1-34  0
[lv_rimage_4] iwi-aor--- linear /dev/sdad:1-34  0
[lv_rimage_5] iwi-aor--- linear /dev/sdae:1-34  0
[lv_rimage_6] iwi-aor--- linear /dev/sdaf:1-34  0
[lv_rmeta_0] ewi-aor--- linear /dev/sda:33-33
[lv_rmeta_1] ewi-aor--- linear /dev/sdaa:33-33
[lv_rmeta_2] ewi-aor--- linear /dev/sdab:33-33
[lv_rmeta_3] ewi-aor--- linear /dev/sdac:0-0
[lv_rmeta_4] ewi-aor--- linear /dev/sdad:0-0
[lv_rmeta_5] ewi-aor--- linear /dev/sdae:0-0
[lv_rmeta_6] ewi-aor--- linear /dev/sdaf:0-0

```

Stripes also can be removed from raid5 and 6. Convert the 5-way striped raid6_nr LV to 4-way-striped. The force option needs to be used, because removing stripes (i.e. image SubLVs) from a RaidLV will shrink its size.

```

# lvconvert --stripes 4 vg/lv
Using default stripesize 64.00 KiB.
WARNING: Removing stripes from active logical volume vg/lv will \
shrink it from 660.00 MiB to 528.00 MiB!
THIS MAY DESTROY (PARTS OF) YOUR DATA!
If that leaves the logical volume larger than 206 extents due \
to stripe rounding,
you may want to grow the content afterwards (filesystem etc.)
WARNING: to remove freed stripes after the conversion has finished,\n
you have to run "lvconvert --stripes 4 vg/lv"
Logical volume vg/lv successfully converted.

```

```

# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
LV      Attr     Type   PE Ranges      DOff
lv      rwi-a-r-s- raid6_nr lv_rimage_0:0-33 \
                  lv_rimage_1:0-33 \
                  lv_rimage_2:0-33 ... \
                  lv_rimage_5:0-33 \
                  lv_rimage_6:0-33  0
[lv_rimage_0] Iwi-aor--- linear /dev/sda:0-32  0
[lv_rimage_0] Iwi-aor--- linear /dev/sda:34-34
[lv_rimage_1] Iwi-aor--- linear /dev/sdaa:0-32  0

```

```
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:34-34
[lv_rimage_2] iwi-aor--- linear /dev/sdab:0-32 0
[lv_rimage_2] iwi-aor--- linear /dev/sdab:34-34
[lv_rimage_3] iwi-aor--- linear /dev/sdac:1-34 0
[lv_rimage_4] iwi-aor--- linear /dev/sdad:1-34 0
[lv_rimage_5] iwi-aor--- linear /dev/sdae:1-34 0
[lv_rimage_6] iwi-aor-R- linear /dev/sdaf:1-34 0
[lv_rmeta_0] ewi-aor--- linear /dev/sda:33-33
[lv_rmeta_1] ewi-aor--- linear /dev/sdaa:33-33
[lv_rmeta_2] ewi-aor--- linear /dev/sdab:33-33
[lv_rmeta_3] ewi-aor--- linear /dev/sdac:0-0
[lv_rmeta_4] ewi-aor--- linear /dev/sdad:0-0
[lv_rmeta_5] ewi-aor--- linear /dev/sdae:0-0
[lv_rmeta_6] ewi-aor-R- linear /dev/sdaf:0-0
```

The 's' in column 9 of the attribute field shows the RaidLV is still reshaping. The 'R' in the same column of the attribute field shows the freed image Sub LVs which will need removing once the reshaping finished.

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
LV Attr Type PE Ranges DOff
lv rwi-a-r-R- raid6_nr lv_rimage_0:0-33 \
    lv_rimage_1:0-33 \
    lv_rimage_2:0-33 ... \
    lv_rimage_5:0-33 \
    lv_rimage_6:0-33 8192
```

Now that the reshape is finished the 'R' attribute on the RaidLV shows images can be removed.

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
LV Attr Type PE Ranges DOff
lv rwi-a-r-R- raid6_nr lv_rimage_0:0-33 \
    lv_rimage_1:0-33 \
    lv_rimage_2:0-33 ... \
    lv_rimage_5:0-33 \
    lv_rimage_6:0-33 8192
```

This is achieved by repeating the command ("lvconvert --stripes 4 vg/lv" would be sufficient).

```
# lvconvert --stripes 4 vg/lv
Using default stripesize 64.00 KiB.
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
LV      Attr   Type   PE Ranges   DOff
lv      rwi-a-r--- raid6_nr lv_rimage_0:0-33 \
    lv_rimage_1:0-33 \
    lv_rimage_2:0-33 ... \
    lv_rimage_5:0-33 8192
[lv_rimage_0] iwi-aor--- linear /dev/sda:0-32 8192
[lv_rimage_0] iwi-aor--- linear /dev/sda:34-34
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:0-32 8192
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:34-34
[lv_rimage_2] iwi-aor--- linear /dev/sdab:0-32 8192
[lv_rimage_2] iwi-aor--- linear /dev/sdab:34-34
```

```
[lv_rimage_3] iwi-aor--- linear /dev/sdac:1-34 8192
[lv_rimage_4] iwi-aor--- linear /dev/sdad:1-34 8192
[lv_rimage_5] iwi-aor--- linear /dev/sdae:1-34 8192
[lv_rmeta_0] ewi-aor--- linear /dev/sda:33-33
[lv_rmeta_1] ewi-aor--- linear /dev/sdaa:33-33
[lv_rmeta_2] ewi-aor--- linear /dev/sdab:33-33
[lv_rmeta_3] ewi-aor--- linear /dev/sdac:0-0
[lv_rmeta_4] ewi-aor--- linear /dev/sdad:0-0
[lv_rmeta_5] ewi-aor--- linear /dev/sdae:0-0
```

```
# lvs -a -o lv_name,attr,segtype,reshapelen vg
LV      Attr  Type   RSize
lv      rwi-a-r--- raid6_nr 24.00m
[lv_rimage_0] iwi-aor--- linear  4.00m
[lv_rimage_0] iwi-aor--- linear
[lv_rimage_1] iwi-aor--- linear  4.00m
[lv_rimage_1] iwi-aor--- linear
[lv_rimage_2] iwi-aor--- linear  4.00m
[lv_rimage_2] iwi-aor--- linear
[lv_rimage_3] iwi-aor--- linear  4.00m
[lv_rimage_4] iwi-aor--- linear  4.00m
[lv_rimage_5] iwi-aor--- linear  4.00m
[lv_rmeta_0] ewi-aor--- linear
[lv_rmeta_1] ewi-aor--- linear
[lv_rmeta_2] ewi-aor--- linear
[lv_rmeta_3] ewi-aor--- linear
[lv_rmeta_4] ewi-aor--- linear
[lv_rmeta_5] ewi-aor--- linear
```

Future developments might include automatic removal of the freed images.

If the reshape space shall be removed any lvconvert command not changing the layout can be used:

```
# lvconvert --stripes 4 vg/lv
Using default stripesize 64.00 KiB.
No change in RAID LV vg/lv layout, freeing reshape space.
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o lv_name,attr,segtype,reshapelen vg
LV      Attr  Type   RSize
lv      rwi-a-r--- raid6_nr  0
[lv_rimage_0] iwi-aor--- linear  0
[lv_rimage_0] iwi-aor--- linear
[lv_rimage_1] iwi-aor--- linear  0
[lv_rimage_1] iwi-aor--- linear
[lv_rimage_2] iwi-aor--- linear  0
[lv_rimage_2] iwi-aor--- linear
[lv_rimage_3] iwi-aor--- linear  0
[lv_rimage_4] iwi-aor--- linear  0
[lv_rimage_5] iwi-aor--- linear  0
[lv_rmeta_0] ewi-aor--- linear
[lv_rmeta_1] ewi-aor--- linear
[lv_rmeta_2] ewi-aor--- linear
[lv_rmeta_3] ewi-aor--- linear
```

```
[lv_rmeta_4] ewi-aor--- linear
[lv_rmeta_5] ewi-aor--- linear
```

In case the RaidLV should be converted to striped:

```
# lvconvert --type striped vg/lv
Unable to convert LV vg/lv from raid6_nr to striped.
Converting vg/lv from raid6_nr is directly possible to the \
following layouts:
    raid6_nc
    raid6_zr
    raid6_la_6
    raid6_ls_6
    raid6_ra_6
    raid6_rs_6
    raid6_n_6
```

A direct conversion isn't possible thus the command informed about the possible ones. raid6_n_6 is suitable to convert to striped so convert to it first (this is a reshape changing the raid6 layout from raid6_nr to raid6_n_6).

```
# lvconvert --type raid6_n_6
Using default stripesize 64.00 KiB.
Converting raid6_nr LV vg/lv to raid6_n_6.
Are you sure you want to convert raid6_nr LV vg/lv? [y/n]: y
Logical volume vg/lv successfully converted.
```

Wait for the reshape to finish.

```
# lvconvert --type striped vg/lv
Logical volume vg/lv successfully converted.
```

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV Attr Type PE Ranges DOff
  lv -wi-a----- striped /dev/sda:2-32 \
    /dev/sdaa:2-32 \
    /dev/sdab:2-32 \
    /dev/sdac:3-33
  lv -wi-a----- striped /dev/sda:34-35 \
    /dev/sdaa:34-35 \
    /dev/sdab:34-35 \
    /dev/sdac:34-35
```

From striped we can convert to raid10

```
# lvconvert --type raid10 vg/lv
Using default stripesize 64.00 KiB.
Logical volume vg/lv successfully converted.
```

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV Attr Type PE Ranges DOff
  lv rwi-a-r--- raid10 lv_rimage_0:0-32 \
    lv_rimage_4:0-32 \
    lv_rimage_1:0-32 ... \
```

```

lv_rimage_3:0-32 \
lv_rimage_7:0-32 0

# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
WARNING: Cannot find matching striped segment for vg/lv_rimage_3.
LV      Attr   Type  PE Ranges    DOff
lv      rwi-a-r--- raid10 lv_rimage_0:0-32 \
                  lv_rimage_4:0-32 \
                  lv_rimage_1:0-32 ... \
                  lv_rimage_3:0-32 \
                  lv_rimage_7:0-32 0
[lv_rimage_0] iwi-aor--- linear /dev/sda:2-32  0
[lv_rimage_0] iwi-aor--- linear /dev/sda:34-35
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:2-32  0
[lv_rimage_1] iwi-aor--- linear /dev/sdaa:34-35
[lv_rimage_2] iwi-aor--- linear /dev/sdab:2-32  0
[lv_rimage_2] iwi-aor--- linear /dev/sdab:34-35
[lv_rimage_3] iwi-XXr--- linear /dev/sdac:3-35  0
[lv_rimage_4] iwi-aor--- linear /dev/sdad:1-33  0
[lv_rimage_5] iwi-aor--- linear /dev/sdae:1-33  0
[lv_rimage_6] iwi-aor--- linear /dev/sdaf:1-33  0
[lv_rimage_7] iwi-aor--- linear /dev/sdag:1-33  0
[lv_rmeta_0] ewi-aor--- linear /dev/sda:0-0
[lv_rmeta_1] ewi-aor--- linear /dev/sdaa:0-0
[lv_rmeta_2] ewi-aor--- linear /dev/sdab:0-0
[lv_rmeta_3] ewi-aor--- linear /dev/sdac:0-0
[lv_rmeta_4] ewi-aor--- linear /dev/sdad:0-0
[lv_rmeta_5] ewi-aor--- linear /dev/sdae:0-0
[lv_rmeta_6] ewi-aor--- linear /dev/sdaf:0-0
[lv_rmeta_7] ewi-aor--- linear /dev/sdag:0-0

```

raid10 allows to add stripes but can't remove them.

A more elaborate example to convert from linear to striped with interim conversions to raid1 then raid5 followed by restripe (4 steps).

We start with the linear LV.

```
# lvs -a -o name,size,segtype,syncpercent,datastripes,\
      stripesize,reshapeable,devices vg
LV  LSize  Type  Cpy%Sync #DStr Stripe RSize Devices
lv  128.00m linear     1   0    /dev/sda(0)
```

Then convert it to a 2-way raid1.

```
# lvconvert --mirrors 1 vg/lv
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o name,size,segtype,datastripes,\
      stripesize,reshapeable,devices vg
LV      LSize  Type  #DStr Stripe RSize Devices
lv      128.00m raid1    2   0    lv_rimage_0(0),\
                  lv_rimage_1(0)
```

```
[lv_rimage_0] 128.00m linear 1 0 /dev/sda(0)
[lv_rimage_1] 128.00m linear 1 0 /dev/sdhx(1)
[lv_rmeta_0] 4.00m linear 1 0 /dev/sda(32)
[lv_rmeta_1] 4.00m linear 1 0 /dev/sdhx(0)
```

Once the raid1 LV is fully synchronized we convert it to raid5_n (only 2-way raid1 LVs can be converted to raid5). We select raid5_n here because it has dedicated parity SubLVs at the end and can be converted to striped directly without any additional conversion.

```
# lvconvert --type raid5_n vg/lv
Using default stripesize 64.00 KiB.
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o name,size,segtype,syncpercent,datastripes,\
      stripesize,reshapelevenle,devices vg
  LV      LSize  Type #DStr Stripe RSize Devices
  lv      128.00m raid5_n 1 64.00k 0 lv_rimage_0(0),\
                  lv_rimage_1(0)
[lv_rimage_0] 128.00m linear 1 0 0 /dev/sda(0)
[lv_rimage_1] 128.00m linear 1 0 0 /dev/sdhx(1)
[lv_rmeta_0] 4.00m linear 1 0 /dev/sda(32)
[lv_rmeta_1] 4.00m linear 1 0 /dev/sdhx(0)
```

Now we'll change the number of data stripes from 1 to 5 and request 128K stripe size in one command. This will grow the size of the LV by a factor of 5 (we add 4 data stripes to the one given). That additional space can be used by e.g. growing any contained filesystem or the LV can be reduced in size after the re-shaping conversion has finished.

```
# lvconvert --stripesize 128k --stripes 5 vg/lv
Converting stripesize 64.00 KiB of raid5_n LV vg/lv to 128.00 KiB.
WARNING: Adding stripes to active logical volume vg/lv will grow \
it from 32 to 160 extents!
Run "lvresize -l32 vg/lv" to shrink it or use the additional capacity.
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o name,size,segtype,datastripes,\
      stripesize,reshapelevenle,devices
  LV      LSize  Type #DStr Stripe RSize Devices
  lv      640.00m raid5_n 5 128.00k 6 lv_rimage_0(0),\
                  lv_rimage_1(0),\
                  lv_rimage_2(0),\
                  lv_rimage_3(0),\
                  lv_rimage_4(0),\
                  lv_rimage_5(0)
[lv_rimage_0] 132.00m linear 1 0 1 /dev/sda(33)
[lv_rimage_0] 132.00m linear 1 0 /dev/sda(0)
[lv_rimage_1] 132.00m linear 1 0 1 /dev/sdhx(33)
[lv_rimage_1] 132.00m linear 1 0 /dev/sdhx(1)
[lv_rimage_2] 132.00m linear 1 0 1 /dev/sdhw(33)
[lv_rimage_2] 132.00m linear 1 0 /dev/sdhw(1)
[lv_rimage_3] 132.00m linear 1 0 1 /dev/sdhv(33)
[lv_rimage_3] 132.00m linear 1 0 /dev/sdhv(1)
[lv_rimage_4] 132.00m linear 1 0 1 /dev/sdhu(33)
[lv_rimage_4] 132.00m linear 1 0 /dev/sdhu(1)
```

```
[lv_rimage_5] 132.00m linear 1 0 1 /dev/sdht(33)
[lv_rimage_5] 132.00m linear 1 0 /dev/sdht(1)
[lv_rmeta_0] 4.00m linear 1 0 /dev/sda(32)
[lv_rmeta_1] 4.00m linear 1 0 /dev/sdhx(0)
[lv_rmeta_2] 4.00m linear 1 0 /dev/sdhw(0)
[lv_rmeta_3] 4.00m linear 1 0 /dev/sdhv(0)
[lv_rmeta_4] 4.00m linear 1 0 /dev/sdhu(0)
[lv_rmeta_5] 4.00m linear 1 0 /dev/sdht(0)
```

Once the conversion has finished we can convert to striped.

```
# lvconvert --type striped vg/lv
Logical volume vg/lv successfully converted.
```

```
# lvs -a -o name,size,segtype,datastripes,\
      stripesize,reshapelen,devices vg
LV LSize Type #DStr Stripe RSize Devices
lv 640.00m striped 5 128.00k /dev/sda(33),\
      /dev/sdhx(33),\
      /dev/sdhw(33),\
      /dev/sdhv(33),\
      /dev/sdhu(33)
lv 640.00m striped 5 128.00k /dev/sda(0),\
      /dev/sdhx(1),\
      /dev/sdhw(1),\
      /dev/sdhv(1),\
      /dev/sdhu(1)
```

Reversing these steps will convert a given striped LV to linear.

Mind the facts that stripes are removed thus the capacity of the RaidLV will shrink and that changing the RaidLV layout will influence its performance.

"lvconvert --stripes 1 vg/lv" for converting to 1 stripe will inform upfront about the reduced size to allow for resizing the content or growing the RaidLV before actually converting to 1 stripe. The **--force** option is needed to allow stripe removing conversions to prevent data loss.

Of course any interim step can be the intended last one (e.g. striped-> raid1).

RAID5 Variants

raid5_ls

- RAID5 left symmetric
- Rotating parity N with data restart

raid5_la

- RAID5 left symmetric
- Rotating parity N with data continuation

raid5_rs

- RAID5 right symmetric
- Rotating parity 0 with data restart

raid5_ra

- RAID5 right asymmetric

- Rotating parity 0 with data continuation

raid5_n

- RAID5 parity n
- Dedicated parity device n used for striped/raid0 conversions
- Used for RAID Takeover

RAID6 Variants

raid6

- RAID6 zero restart (aka left symmetric)
- Rotating parity 0 with data restart
- Same as raid6_zr

raid6_zr

- RAID6 zero restart (aka left symmetric)
- Rotating parity 0 with data restart

raid6_nr

- RAID6 N restart (aka right symmetric)
- Rotating parity N with data restart

raid6_nc

- RAID6 N continue
- Rotating parity N with data continuation

raid6_n_6

- RAID6 last parity devices
- Fixed dedicated last devices (P–Syndrome N–1 and Q–Syndrome N) with striped data used for striped/raid0 conversions
- Used for RAID Takeover

raid6_{ls,rs,la,ra}_6

- RAID6 last parity device
- Dedicated last parity device used for conversions from/to raid5_{ls,rs,la,ra}

raid6_ls_6

- RAID6 N continue
- Same as raid5_ls for N–1 devices with fixed Q–Syndrome N
- Used for RAID Takeover

raid6_la_6

- RAID6 N continue
- Same as raid5_la for N–1 devices with fixed Q–Syndrome N
- Used for RAID Takeover

raid6_rs_6

- RAID6 N continue
- Same as raid5_rs for N–1 devices with fixed Q–Syndrome N
- Used for RAID Takeover

raid6_ra_6

- RAID6 N continue
- Same as raid5_ra for N–1 devices with fixed Q–Syndrome N

- Used for RAID Takeover

History

The 2.6.38-rc1 version of the Linux kernel introduced a device-mapper target to interface with the software RAID (MD) personalities. This provided device-mapper with RAID 4/5/6 capabilities and a larger development community. Later, support for RAID1, RAID10, and RAID1E (RAID 10 variants) were added. Support for these new kernel RAID targets was added to LVM version 2.02.87. The capabilities of the LVM **raid1** type have surpassed the old **mirror** type. **raid1** is now recommended instead of **mirror**. **raid1** became the default for mirroring in LVM version 2.02.100.

NAME

`lvmreport` — LVM reporting and related features

DESCRIPTION

LVM uses single reporting infrastructure that sets standard on LVM command's output and it provides wide range of configuration settings and command line options to customize report and filter the report's output.

Categorization based on reporting facility

Based on functionality, commands which make use of the reporting infrastructure are divided in two groups:

Report-oriented

These commands inform about current LVM state and their primary role is to display this information in compendious way. To make a distinction, we will name this report as **main report**. The set of report-only commands include: `pvs`, `vgs`, `lvs`, `pvdisplay`, `vgdisplay`, `lvdisplay`, `lvm devtypes`, `lvm fullreport`. For further information about main report, see **main report specifics**.

Processing-oriented

These commands are responsible for changing LVM state and they do not contain any main report as identified for report-oriented commands, they only perform some kind of processing. The set of processing-oriented commands includes: `pvccreate`, `vgcreate`, `lvcreate`, `pvchange`, `vgchange`, `lvchange`, `pvremove`, `vgremove`, `lvremove`, `pvresize`, `vgextend`, `vgreduce`, `lvextend`, `lvreduce`, `lvresize`, `lvrename`, `pvscan`, `vgscan`, `lvscan`, `pvmove`, `vgcfgbackup`, `vgck`, `vgconvert`, `vgexport`, `vgimport`, `vgmknodes`.

If enabled, so called **log report** is either displayed solely (for processing-oriented commands) or in addition to main report (for report-oriented commands). The log report contains a log of operations, messages and per-object status with complete object identification collected during LVM command execution. See **log report specifics** for more information about this report type.

Terms

When describing reporting functionality and features in this text, we will use terms **row** and **column**. By row we mean series of values reported for single entity (for example single PV, VG or LV). Each value from the row then belongs to a column of certain type. The columns have **column headings** which are short descriptions for the columns. The columns are referenced by **column names**. Please note that this text is also using term **field** interchangeably with the term **column**. Most of the time the term columns is abbreviated as **col** in configuration.

Common report configuration settings and command line options

There are common configuration settings and command line options which apply to both **main report** and **log report**. Following lists contain all of them, separated into groups based on their use.

Common configuration settings:

- Changing report output format, composition and other output modifiers:
 - `global/units`
 - `global/suffix`
 - `report/output_format`
 - `report/compact_output`

- report/compact_output_cols
- report/aligned
- report/headings
- report/separator
- report/list_item_separator
- report/prefixes
- report/quoted
- report/columns_as_rows
- report/binary_values_as_numeric
- report/time_format
- report/mark_hidden_devices
- report/two_word_unknown_device
- Special settings
 - report/buffered

This document does not describe these settings in more detail – if you need detailed information, including values which are accepted for the settings, please run **lvmconfig --type default --withcomments <setting>**. There are more configuration settings in addition to the common set listed above, but they are specific to either **main report** or **log report**, see **main report specifics** and **log report specifics** for these settings. Besides configuring reports globally by using configuration settings, there are also command line options you can use to extend, override or further specify the report configuration.

Common command line options:

- Definition of the set set of fields to use
 - --options|-o FieldSet

Field set to use. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option overrides.
 - --options|-o+ FieldSet

Fields to include to current field set. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option extends.
 - --options|-o- FieldSet

Fields to exclude from current field set. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option reduces.
 - --options|-o# FieldSet

Compaction of unused fields. Overrides report/compact_output_cols configuration setting.
- Sorting

- --sort|–O+ FieldSet
Fields to sort by in ascending order. See **main report specifics** and **log report specifics** for information about field sets configured with global configuration settings that this option overrides.
- --sort|–O- FieldSet
Fields to sort by in descending order. See **main report specifics** and **log report specifics** for information about fields sets configured with global configuration settings that this options overrides.
- Selection
 - --select|–S Selection
Define selection criteria for report output. For **log report**, this also overrides log/command_log_selection configuration setting, see also **log report specifics**.
- Changing output format and composition
 - --reportformat
Overrides report/output_format configuration setting.
 - --aligned
Overrides report/aligned configuration setting.
 - --binary
Overrides report/binary_values_as_numeric configuration setting.
 - --nameprefixes
Overrides report/prefixes configuration setting.
 - --noheadings
Overrides report/noheadings configuration setting.
 - --nosuffix
Overrides global/suffix configuration setting.
 - --rows
Overrides report/columns_as_rows configuration setting.
 - --separator
Overrides report/separator configuration setting.
 - --units
Overrides global/units configuration setting.
 - --unquoted
Overrides report/quoted configuration setting.
- Special options
 - --configreport **ReportName**
This defines the **ReportName** for which any subsequent –o--columns, -O--sort or –S--select applies to. See also **main report specifics** and **log report specifics** for possible **ReportName** values.
 - --logonly
When an LVM command contains both **main report** and **log report**, this option suppresses the **main report** output and it causes the **log report** output to be displayed only.
 - --unbuffered
Overrides report/buffered configuration setting.

The **FieldSet** mentioned in the lists above is a set of field names where each field name is delimited by "," character. Field set definition, sorting and selection may be repeated on command line (-o+/-o- includes/excludes fields to/from current list, for all the other repeatable options, the last value typed for the option on the command line is used). The **Selection** is a string with **selection criteria**, see also **Selection** paragraph below for more information about constructing these criteria.

Main report specifics

The **main report** currently encompasses these distinct subtypes, referenced by their name - **ReportName** as listed below. The command in parenthesis is representative command that uses the main report subtype by default. Each subtype has its own configuration setting for global field set definition as well as sort field definition (listed below each individual **ReportName**):

- **pv** representing report about Physical Volumes (**pvs**)
 - report/pvs_cols
 - report/pvs_sort
- **pvseg** representing report about Physical Volume Segments (**pvs --segments**)
 - report/pvseg_cols
 - report/pvseg_sort
- **vg** representing report about Volume Groups (**vgs**)
 - report/vgs_cols
 - report/vgs_sort
- **lv** representing report about Logical Volumes (**lvs**)
 - report/lvs_cols
 - report/lvs_sort
- **seg** representing report about Logical Volume Segments (**lvs --segments**)
 - report/segs_cols
 - report/segs_sort
- **full** representing report combining all of the above as a whole (**lvm fullreport**)
 - report/pvs_cols_full
 - report/pvs_sort_full
 - report/pvsegs_cols_full
 - report/pvseg_sort_full
 - report/vgs_cols_full
 - report/vgs_sort_full
 - report/lvs_cols_full
 - report/lvs_sort_full

- report/segs_cols_full
- report/segs_sort_full
- **devtype** representing report about device types (**lvm devtypes**)
 - report/devtypes_cols
 - report/devtypes_sort

Use **pvs**, **vgs**, **lvs -o help** or **lvm devtypes -o help** to get complete list of fields that you can use for main report. The list of fields in the help output is separated in groups based on which report type they belong to. Note that LVM can change final report type used if fields from different groups are combined together. Some of these combinations are not allowed in which case LVM will issue an error.

For all main report subtypes except **full**, it's not necessary to use **--configreport ReportName** to denote which report any subsequent **-o**, **-O** or **-S** option applies to as they always apply to the single main report type. Currently, **lvm fullreport** is the only command that includes more than one **main report** subtype. Therefore, the **--configreport** is particularly suitable for the full report if you need to configure each of its subreports in a different way.

Log report specifics

You can enable log report with **log/report_command_log** configuration setting – this functionality is disabled by default. The **log report** contains a log collected during LVM command execution and then the log is displayed just like any other report known from main report. There is only one log report subtype as shown below together with related configuration settings for fields, sorting and selection:

- **log** representing log report
 - log/command_log_cols
 - log/command_log_sort
 - log/command_log_selection

You always need to use **--configreport log** together with **-o--options**, **-O--sort** or **-S--selection** to override configuration settings directly on command line for **log report**. When compared to **main report**, in addition to usual configuration settings for report fields and sorting, the **log report** has also configuration option for selection - **report/command_log_selection**. This configuration setting is provided for convenience so it's not necessary to use **-S--select** on command line each time an LVM command is executed and we need the same selection criteria to be applied for **log report**. Default selection criteria used for **log report** are **log/command_log_selection="!(log_type=status && message=success)"**. This means that, by default, **log report** doesn't display status messages about successful operation and it displays only rows with error, warning, print-type messages and messages about failure states (for more information, see **log report content** below).

Log report coverage

Currently, when running LVM commands directly (not in LVM shell), the log report covers command's **processing stage** which is the moment when LVM entities are iterated and processed one by one. It does not cover any command initialization nor command finalization stage. If there is any message issued out of log report's coverage range, such message goes directly to output, bypassing the **log report**. By default, that is **standard error output** for error and warning messages and **standard output** for common print-like messages.

When running LVM commands in **LVM shell**, the log report covers the whole LVM command's execution, including command's **processing** as well as **initialization** and **finalization stage**. So from this point of view, the log report coverage is complete for executed LVM commands. Note that there are still a few moments when LVM shell needs to initialize itself before it even enters the main loop in which it executes LVM commands. Also, there is a moment when **LVM shell** needs to prepare **log report** properly for next command executed in the shell and then, after the command's run, the shell needs to display the log report for that recently executed command. If there is a failure or any other message issued during this time, the LVM will bypass **log report** and display messages on output directly.

For these reasons and for completeness, it's not possible to rely fully on **log report** as the only indicator of LVM command's status and the only place where all messages issued during LVM command execution are collected. You always need to check whether the command has not failed out of log report's range by checking the non-report output too.

To help with this, LVM can separate output which you can then redirect to any **custom file descriptor** that you prepare before running an LVM command or LVM shell and then you make LVM to use these file descriptors for different kinds of output by defining environment variables with file descriptor numbers. See also **LVM_OUT_FD**, **LVM_ERR_FD** and **LVM_REPORT_FD** environment variable description in **lvm(8)** man page.

Also note that, by default, reports use the same file descriptor as common print-like messages, which is **standard output**. If you plan to use **log report** in your scripts or any external tool, you should use **LVM_OUT_FD**, **LVM_ERR_FD** and **LVM_REPORT_FD** to separate all output types to different file descriptors. For example, with bash, that would be:

```
LVM_OUT_FD=3 LVM_ERR_FD=4 LVM_REPORT_FD=5 <lvm command> 3>out_file
4>err_file 5>report_file
```

Where the <lvm_command> is either direct LVM command or LVM shell. You can collect all three types of output in particular files then.

Log report content

Each item in the log report consists of these set of fields providing various information:

- Basic information (mandatory):
 - **log_seq_num**
Item sequence number. The sequence number is unique for each log item and it increases in the order of the log items as they appeared during LVM command execution.
 - **log_type**
Type of log for the item. Currently, these types are used:
 - status** for any status information that is logged
 - print** for any common message printed while the log is collected
 - error** for any error message printed while the log is collected
 - warn** for any warning message printed while the log is collected
 - **log_context**
Context of the log for the item. Currently, two contexts are identified:
 - shell** for the log collected in the outermost code before and after executing concrete LVM commands

processing for the log collected while processing LVM entities during LVM command execution

- Message (mandatory):
 - **log_message**
Any message associated with current item. For **status** log type, the message contains either **success** or **failure** denoting current state. For **print**, **error** and **warn** log types, the message contains the exact message of that type that got issued.
- Object information (used only if applicable):
 - **log_object_type** field
Type of the object processed. Currently, these object types are recognized:
 - cmd** for command as a whole
 - orphan** for processing group of PVs not in any VG yet
 - pv** for PV processing
 - label** for direct PV label processing (without VG metadata)
 - vg** for VG processing
 - lv** for LV processing
 - **log_object_name**
Name of the object processed.
 - **log_object_id**
ID of the object processed.
 - **log_object_group**
A group where the processed object belongs to.
 - **log_object_group_id**
An ID of a group where the processed object belongs to.
- Numeric status (used only if applicable)
 - **log_errno**
Error number associated with current item.
 - **log_ret_code**
Return code associated with current item.

You can also run **<lvm_command> --configreport log -o help** to display complete list of fields that you may use for the **log report**.

Selection

Selection is used for a report to display only rows that match **selection criteria**. All rows are displayed with the additional **selected** field (**-o selected**) displaying 1 if the row matches the *Selection* and 0 otherwise. The **selection criteria** are a set of **statements** combined by **logical and grouping operators**. The

statement consists of a **field** name for which a set of valid **values** is defined using **comparison operators**. For complete list of fields names that you can use in selection, see the output of **<lvm_command> -S help**. The help output also contains type of values that each field displays enclosed in brackets.

List of operators recognized in selection criteria

- Comparison operators (cmp_op)
 - =~ matching regular expression.
 - !~ not matching regular expression.
 - = equal to.
 - != not equal to.
 - >= greater than or equal to.
 - > greater than
 - <= less than or equal to.
 - < less than.
- Binary logical operators (cmp_log)
 - && all fields must match
 - , all fields must match
 - || at least one field must match
 - # at least one field must match
- Unary logical operators
 - ! logical negation
- Grouping operators
 - (left parenthesis
 -) right parenthesis
 - [list start
 -] list end
 - { list subset start
 - } list subset end

Field types and selection operands

Field type restricts the set of operators and values that you may use with the field when defining selection criteria. You can see field type for each field if you run **<lvm command> -S help** where you can find the type name enclosed in square brackets. Currently, LVM recognizes these field types in reports:

- **string** for set of characters (for each string field type, you can use either string or regular expression – regex for the value used in selection criteria)
- **string list** for set of strings
- **number** for integer value
- **size** for integer or floating point number with size unit suffix (see also **lvcreate(8)** man page and description for “**-L--size**” option for the list of recognized suffixes)

- **percent** for floating point number with or without "%" suffix (e.g. 50 or 50%)
- **time** for time values

When using **string list** in selection criteria, there are several ways how LVM can match string list fields from report, depending on what list grouping operator is used and what item separator is used within that set of items. Also, note that order of items does not matter here.

- **matching the set strictly** where all items must match – use [], e.g. ["a","b","c"]
- **matching a subset of the set** – use { } with "," or "&&&" as item delimiter, e.g. {"a","b","c"}
- **matching an intersection with the set** – use { } with "#" or "||" as item delimiter, e.g. {"a" || "b" || "c"}

When using **time** in your selection criteria, LVM can recognize various time formats using standard, absolute or freeform expressions. For examples demonstrating time expressions in selection criteria, see **EXAMPLES** section.

- **Standard time format**

- date

YYYY-MM-DD

YYYY-MM, auto DD=1

YYYY, auto MM=01 and DD=01

- time

hh:mm:ss

hh:mm, auto ss=0

hh, auto mm=0, auto ss=0

- timezone

+hh:mm or -hh:mm

+hh or -hh

The full date/time specification is YYYY-MM-DD hh:mm:ss. Users are able to leave date/time parts from right to left. Whenever these parts are left out, a range is assumed automatically with second granularity. For example:

"2015-07-07 9:51" means range of "2015-07-07 9:51:00" - "2015-07-07 9:51:59".

"2015-07" means range of "2015-07-01 0:00:00" - "2015-07-31 23:59:59"

"2015" means range of "2015-01-01 0:00:00" - "2015-12-31 23:59:59"

- **Absolute time format**

Absolute time is defined as number of seconds since the Epoch (1970:01:01 00:00 +00:00).

- @seconds
- **Freeform time format**
 - weekday names ("Sunday" - "Saturday" or abbreviated as "Sun" - "Sat")
 - labels for points in time ("noon", "midnight")
 - labels for a day relative to current day ("today", "yesterday")
 - points back in time with relative offset from today (N is a number)
 - "N" "seconds" / "minutes" / "hours" / "days" / "weeks" / "years" "ago"
 - "N" "secs" / "mins" / "hrs" ... "ago"
 - "N" "s" / "m" / "h" ... "ago"
 - time specification either in hh:mm:ss format or with AM/PM suffixes
 - month names ("January" - "December" or abbreviated as "Jan" - "Dec")

Informal grammar specification

STATEMENT = column cmp_op **VALUE** | **STATEMENT** log_op **STATEMENT** | (**STATEMENT**) | !(**STATEMENT**)

VALUE = [**VALUE** log_op **VALUE**]

For list-based types: string list. Matches strictly. The log_op must always be of one type within the whole list value.

VALUE = { **VALUE** log_op **VALUE** }

For list-based types: string list. Matches a subset. The log_op must always be of one type within the whole list value.

VALUE = **value**

For scalar types: number, size, percent, string (or string regex).

EXAMPLES

Basic usage

We start our examples with default configuration - **lvmconfig(8)** is helpful command to display configuration settings which are currently used, including all configuration related to reporting. We will use it throughout examples below to display current configuration.

```
# lvmconfig --type full global/global/suffix \
report/output_format report/compact_output \
report/compact_output_cols report/aligned \
report/headings report/sePARATOR \
report/list_item_separator report/prefixes \
report/quoted report/columns_as_rows \
report/binary_values_as_numeric report/time_format \
report/mark_hidden_devices report/two_word_unknown_device \
report/buffered
units="h"
suffix=1
output_format="basic"
compact_output=0
compact_output_cols=""
aligned=1
headings=1
```

```

separator=" "
list_item_separator ","
prefixes=0
quoted=1
columns_as_rows=0
binary_values_as_numeric=0
time_format="%Y-%m-%d %T %z"
mark_hidden_devices=1
two_word_unknown_device=0
buffered=1

```

Also, we start with simple LVM layout with two PVs (/dev/sda, /dev/sdb), VG (vg) and two LVs (lvol0 and lvol1) in the VG. We display all possible reports as single commands here, see also **pvs(8)**, **vgs(8)**, **lvs(8)** man pages for more information. The field set for each report type is configured with configuration settings as we already mentioned in **main report specifics** section in this man page.

```

# lvmconfig --type full report/pvs_cols report/pvs_sort \
    report/pvsegs_cols report/pvsegs_sort report/vgs_cols \
    report/vgs_sort report/lvs_cols report/lvs_sort \
    report/segs_cols report/segs_sort
pvs_cols="pv_name,vg_name,pv_fmt,pv_attr,pv_size,pv_free"
pvs_sort="pv_name"
pvsegs_cols="pv_name,vg_name,pv_fmt,pv_attr,pv_size,pv_free,
            pvseg_start,pvseg_size"
pvsegs_sort="pv_name,pvseg_start"
vgs_cols="vg_name,pv_count,lv_count,snap_count,vg_attr,vg_size,vg_free"
vgs_sort="vg_name"
lvs_cols="lv_name,vg_name,lv_attr,lv_size,pool_lv,origin,move_pv,
         mirror_log,copy_percent,convert_lv"
lvs_sort="vg_name,lv_name"
segs_cols="lv_name,vg_name,lv_attr,stripes,segtype,seg_size"
segs_sort="vg_name,lv_name,seg_start"

# pvs
PV      VG Fmt Attr PSize  PFree
/dev/sda  vg lvm2 a-- 100.00m 88.00m
/dev/sdb  vg lvm2 a-- 100.00m 92.00m

# pvs --segments
PV      VG Fmt Attr PSize  PFree Start SSize
/dev/sda  vg lvm2 a-- 100.00m 88.00m  0   1
/dev/sda  vg lvm2 a-- 100.00m 88.00m  1   1
/dev/sda  vg lvm2 a-- 100.00m 88.00m  2   1
/dev/sda  vg lvm2 a-- 100.00m 88.00m  3   22
/dev/sdb  vg lvm2 a-- 100.00m 92.00m  0   1
/dev/sdb  vg lvm2 a-- 100.00m 92.00m  1   1
/dev/sdb  vg lvm2 a-- 100.00m 92.00m  2   23

# vgs
VG #PV #LV #SN Attr  VSize  VFree
vg 2 2 0 wz--n- 200.00m 180.00m

# lvs
LV  VG Attr     LSize Pool Origin Move Log Cpy%Sync Convert

```

```

lvol0 vg -wi-a----- 4.00m
lvol1 vg rwi-a-r--- 4.00m          100.00

# lvs --segments
LV  VG Attr #Str Type SSize
lvol0 vg -wi-a----- 1 linear 4.00m
lvol1 vg rwi-a-r--- 2 raid1 4.00m

```

We will use **report/lvs_cols** and **report/lvs_sort** configuration settings to define our own list of fields to use and to sort by that is different from defaults. You can do this for other reports in same manner with **report/{pvs,pvseg,vgs,seg}_{cols,sort}** configuration settings. Also note that in the example below, we don't display the "lv_time" field even though we're using it for sorting – this is allowed.

```
# lvmconfig --type full report/lvs_cols report/lvs_sort
lvs_cols="lv_name,lv_size,origin,pool_lv,copy_percent"
lvs_sort="-lv_time"
```

```
# lvs
LV  LSize Origin Pool Cpy%Sync
lvol1 4.00m      100.00
lvol0 4.00m
```

You can use **-o--options** command line option to override current configuration directly on command line.

```
# lvs -o lv_name,lv_size
LV  LSize
lvol1 4.00m
lvol0 4.00m

# lvs -o+lv_layout
LV  LSize Origin Pool Cpy%Sync Layout
lvol1 4.00m      100.00  raid,raid1
lvol0 4.00m      linear

# lvs -o-origin
LV  LSize Pool Cpy%Sync
lvol1 4.00m    100.00
lvol0 4.00m

# lvs -o lv_name,lv_size,origin -o+lv_layout -o-origin -O lv_name
LV  LSize Layout
lvol0 4.00m linear
lvol1 4.00m raid,raid1
```

You can obtain the same information with single command where all the information about PVs, PV segments, LVs and LV segments are obtained per VG under a single VG lock for consistency, see also **lvm-fullreport(8)** man page for more information. The fullreport has its own configuration settings to define field sets to use, similar to individual reports as displayed above, but configuration settings have "_full" suffix now. This way, it's possible to configure different sets of fields to display and to sort by for individual reports as well as the full report.

```
# lvmconfig --type full report/pvs_cols_full \
report/pvs_sort_full report/pvsegs_cols_full \
report/pvsegs_sort_full report/vgs_cols_full \
```

```

report/vgs_sort_full report/lvs_cols_full \
report/lvs_sort_full report/segs_cols_full \
report/segs_sort_full
pvs_cols_full="pv_name,vg_name"
pvs_sort_full="pv_name"
pvsegs_cols_full="pv_name,pvseg_start,pvseg_size"
pvsegs_sort_full="pv_uuid,pvseg_start"
vgs_cols_full="vg_name"
vgs_sort_full="vg_name"
lvs_cols_full="lv_name,vg_name"
lvs_sort_full="vg_name,lv_name"
segs_cols_full="lv_name,seg_start,seg_size"
segs_sort_full="lv_uuid,seg_start"

# lvm fullreport
VG
vg
PV      VG
/dev/sda  vg
/dev/sdb  vg
LV      VG
lvol0 vg
lvol1 vg
PV      Start SSize
/dev/sda  0   1
/dev/sda  1   1
/dev/sda  2   1
/dev/sda  3   22
/dev/sdb  0   1
/dev/sdb  1   1
/dev/sdb  2   23
LV      Start SSize
lvol0  0  4.00m
lvol1  0  4.00m

```

Automatic output compaction

If you look at the lvs output above, you can see that the report also contains fields for which there is no information to display (e.g. the columns under "Origin" and "Pool" heading – the "origin" and "pool_lv" fields). LVM can automatically compact report output so such fields are not included in final output. To enable this feature and to compact all fields, use **report/compact_output=1** in your configuration.

```
# lvmconfig --type full report/compact_output
compact_output=1
```

```

# lvs
LV  LSize Cpy%Sync
lvol1 4.00m 100.00
lvol0 4.00m

# lvs vg/lvol0
LV  LSize
lvol0 4.00m

```

Alternatively, you can define which fields should be compacted by configuring

report/compact_output_cols configuration setting (or **-o--options** # command line option).

```
# lvmconfig --type full report/compact_output report/compact_output_cols
compact_output=0
compact_output_cols="origin"

# lvs
LV  LSize Pool Cpy%Sync
lvol1 4.00m    100.00
lvol0 4.00m

# lvs vg/lvol0
LV  LSize Pool
lvol0 4.00m

# lvs -o#pool_lv
LV  LSize Origin Cpy%Sync
lvol1 4.00m    100.00
lvol0 4.00m
```

We will use **report/compact_output=1** for subsequent examples.

Further formatting options

By default, LVM displays sizes in reports in human-readable form which means that the most suitable unit is used so it's easy to read. You can use **report/units** configuration setting (or **--units** option directly on command line) and **report/suffix** configuration setting (or **--nosuffix** command line option) to change this.

```
# lvs --units b --nosuffix
LV  LSize  Cpy%Sync
lvol1 4194304 100.00
lvol0 4194304
```

If you want to configure whether report headings are displayed or not, use **report/headings** configuration settings (or **--noheadings** command line option).

```
# lvs --noheadings
lvol1 4.00m 100.00
lvol0 4.00m
```

In some cases, it may be useful to display report content as key=value pairs where key here is actually the field name. Use **report/prefixes** configuration setting (or **--nameprefixes** command line option) to switch between standard output and the key=value output. The key=value pair is the output that is suitable for use in scripts and for other tools to parse easily. Usually, you also don't want to display headings with the output that has these key=value pairs.

```
# lvs --noheadings --nameprefixes
LVM2_LV_NAME='lvol1' LVM2_LV_SIZE='4.00m' LVM2_COPY_PERCENT='100.00'
LVM2_LV_NAME='lvol0' LVM2_LV_SIZE='4.00m' LVM2_COPY_PERCENT=""
```

To define whether quotation marks in key=value pairs should be used or not, use **report/quoted** configuration setting (or **--unquoted** command line option).

```
# lvs --noheadings --nameprefixes --unquoted
LVM2_LV_NAME=lvol1 LVM2_LV_SIZE=4.00m LVM2_COPY_PERCENT=100.00
```

```
LVM2_LV_NAME=lvol0 LVM2_LV_SIZE=4.00m LVM2_COPY_PERCENT=
```

For easier parsing, you can even transpose the report so each column now becomes a row in the output. This is done with **report/output_as_rows** configuration setting (or **--rows** command line option).

```
# lvs --noheadings --nameprefixes --unquoted --rows
LVM2_LV_NAME=lvol1 LVM2_LV_NAME=lvol0
LVM2_LV_SIZE=4.00m LVM2_LV_SIZE=4.00m
LVM2_COPY_PERCENT=100.00 LVM2_COPY_PERCENT=
```

Use **report/separator** configuration setting (or **--separator** command line option) to define your own field separator to use.

```
# lvs --noheadings --nameprefixes --unquoted --separator " | "
LVM2_LV_NAME=lvol1 | LVM2_LV_SIZE=4.00m | LVM2_COPY_PERCENT=100.00
LVM2_LV_NAME=lvol0 | LVM2_LV_SIZE=4.00m | LVM2_COPY_PERCENT=
```

If you are using your own separator, the columns in the output are not aligned by default. Use **report/aligned** configuration setting (or **--aligned** command line option) for LVM to add extra spaces in report to align the output properly.

```
# lvs --separator " | "
LV | LSize | Cpy%Sync
lvol1 | 4.00m | 100.00
lvol0 | 4.00m |
```

```
# lvs --separator " | " --aligned
LV | LSize | Cpy%Sync
lvol1 | 4.00m | 100.00
lvol0 | 4.00m |
```

Let's display one more field in addition ("lv_tags" in this example) for the lvs report output.

```
# lvs -o+lv_tags
LV  LSize Cpy%Sync LV Tags
lvol1 4.00m 100.00
lvol0 4.00m      tagA,tagB
```

The "LV Tags" column in the example above displays two list values, separated by "," character for LV lvol0. If you need different list item separator, use **report/list_item_separator** configuration setting its definition.

```
# lvmconfig --type full report/list_item_separator
list_item_separator=";"

# lvs -o+tags
LV  LSize Cpy%Sync LV Tags
lvol1 4.00m 100.00
lvol0 4.00m      tagA;tagB
```

But let's still use the original "," character for list_item_separator for subsequent examples.

Format for any of time values displayed in reports can be configured with **report/time_format** configuration setting. By default complete date and time is displayed, including timezone.

```
# lvmconfig --type full report/time_format
time_format="%Y-%m-%d %T %z"

# lvs -o+time
LV  LSize Cpy%Sync CTime
lvol1 4.00m 100.00 2016-08-29 12:53:36 +0200
lvol0 4.00m      2016-08-29 10:15:17 +0200
```

We can change time format in similar way as we do when using **date(1)** command or **strftime(3)** function (**lvmconfig --type default --withcomments report/time_format** will give you complete list of available formatting options). In the example below, we decided to use **%s** for number of seconds since Epoch (1970-01-01 UTC).

```
# lvmconfig --type full report/time_format
time_format="%s"

# lvs
LV Attr LSize Cpy%Sync LV Tags CTime
lvol1 rwi-a-r--- 4.00m 100.00      1472468016
lvol0 -wi-a----- 4.00m          tagA,tagB 1472458517
```

The **lvs** does not display hidden LVs by default – to include these LVs in the output, you need to use **-a--all** command line option. Names for these hidden LVs are displayed within square brackets.

```
# lvs -a
LV      LSize Cpy%Sync
lvol1    4.00m 100.00
[lvol1_rimage_0] 4.00m
[lvol1_rmeta_0] 4.00m
[lvol1_rimage_1] 4.00m
[lvol1_rmeta_1] 4.00m
lvol0    4.00m
```

You can configure LVM to display the square brackets for hidden LVs or not with **report/mark_hidden_devices** configuration setting.

```
# lvmconfig --type full report/mark_hidden_devices
mark_hidden_devices=0

# lvs -a
LV      LSize Cpy%Sync
lvol1    4.00m 100.00
lvol1_rimage_0 4.00m
lvol1_rmeta_0 4.00m
lvol1_rimage_1 4.00m
lvol1_rmeta_1 4.00m
lvol0    4.00m
```

It's not recommended to use LV marks for hidden devices to decide whether the LV is the one to use by end users or not. Please, use "lv_role" field instead which can report whether the LV is "public" or "private". The private LVs are used by LVM only and they should not be accessed directly by end users.

```
# lvs -a -o+lv_role
LV      LSize Cpy%Sync Role
```

```

lvol1      4.00m 100.00  public
lvol1_rimage_0 4.00m      private,raid,image
lvol1_rmeta_0 4.00m      private,raid,metadata
lvol1_rimage_1 4.00m      private,raid,image
lvol1_rmeta_1 4.00m      private,raid,metadata
lvol0      4.00m      public

```

Some of the reporting fields that LVM reports are of binary nature. For such fields, it's either possible to display word representation of the value (this is used by default) or numeric value (0/1 or -1 in case the value is undefined).

```

# lvs -o+lv_active_locally
LV  LSize Cpy%Sync ActLocal
lvol1 4.00m 100.00  active locally
lvol0 4.00m      active locally

```

We can change the way how these binary values are displayed with **report/binary_values_as_numeric** configuration setting.

```

# lvmconfig --type full report/binary_values_as_numeric
binary_values_as_numeric=1

# lvs -o+lv_active_locally
LV  LSize Cpy%Sync ActLocal
lvol1 4.00m 100.00      1
lvol0 4.00m            1

```

Changing output format

LVM can output reports in different formats – use **report/output_format** configuration setting (or **--reportformat** command line option) to switch the report output format. Currently, LVM supports "**basic**" (all the examples we used above used this format) and "**JSON**" output format.

```

# lvs -o lv_name,lv_size --reportformat json
{
  "report": [
    {
      "lv": [
        {"lv_name": "lvol1", "lv_size": "4.00m"},  

        {"lv_name": "lvol0", "lv_size": "4.00m"}
      ]
    }
  ]
}

```

Note that some configuration settings and command line options have no effect with certain report formats. For example, with **JSON** output, it doesn't have any meaning to use **report/aligned** (**--aligned**), **report/noheadings** (**--noheadings**), **report/columns_as_rows** (**--rows**) or **report/buffered** (**--unbuffered**). All these configuration settings and command line options are ignored if using the **JSON** report output format.

Selection

If you need to select only specific rows from report, you can use LVM's report selection feature. If you call **<lvmt command> -S help**, you'll get quick help on selection. The help contains list of all fields that LVM can use in reports together with its type enclosed in square brackets. The example below contains a line

from lvs -S help.

```
# lvs -S help
...
lv_size      - Size of LV in current units. [size]
...
```

This line tells you that the "lv_size" field is of "size" type. If you look at the bottom of the help output, you can see section about "Selection operators" and its "Comparison operators".

```
# lvs -S help
...
Selection operators
-----
Comparison operators:
=~= - Matching regular expression. [regex]
!~= - Not matching regular expression. [regex]
= - Equal to. [number, size, percent, string, string_list, time]
!= - Not equal to. [number, size, percent, string, string_list, time]
>= - Greater than or equal to. [number, size, percent, time]
> - Greater than. [number, size, percent, time]
<= - Less than or equal to. [number, size, percent, time]
< - Less than. [number, size, percent, time]
since - Since specified time (same as '>='). [time]
after - After specified time (same as '>'). [time]
until - Until specified time (same as '<='). [time]
before - Before specified time (same as '<'). [time]
...

```

Here you can match comparison operators that you may use with the "lv_size" field which is of type "size" - it's =, !=, >=, >, <= and <. You can find applicable comparison operators for other fields and other field types the same way.

To demonstrate selection functionality in LVM, we will create more LVs in addition to lvol0 and lvol1 we used in our previous examples.

```
# lvs -o name,size,origin,snap_percent,tags,time
LV  LSize Origin Snap% LV Tags      CTime
lvol4 4.00m lvol2 24.61      2016-09-09 16:57:44 +0200
lvol3 4.00m lvol2 5.08      2016-09-09 16:56:48 +0200
lvol2 8.00m          tagA,tagC,tagD 2016-09-09 16:55:12 +0200
lvol1 4.00m          2016-08-29 12:53:36 +0200
lvol0 4.00m          tagA,tagB   2016-08-29 10:15:17 +0200
```

When selecting size and percent fields, we don't need to use units. For sizes, default "m" (for MiB) is used – this is the same behaviour as already used for LVM commands when specifying sizes (e.g. lvcreate -L). For percent fields, "%" is assumed automatically if it's not specified. The example below also demonstrates how several criteria can be combined together.

```
# lvs -o name,size,snap_percent -S 'size=8m'
LV  LSize
lvol2 8.00m
```

```
# lvs -o name,size,snap_percent -S 'size=8'
```

```

LV  LSize
lvol2 8.00m

# lvs -o name,size,snap_percent -S 'size < 5000k'
LV  LSize Snap%
lvol4 4.00m 24.61
lvol3 4.00m 5.08
lvol1 4.00m
lvol0 4.00m

# lvs -o name,size,snap_percent -S 'size < 5000k && snap_percent > 20'
LV  LSize Snap%
lvol4 4.00m 24.61

# lvs -o name,size,snap_percent \
-S '(size < 5000k && snap_percent > 20%) || name=lvol2'
LV  LSize Snap%
lvol4 4.00m 24.61
lvol2 8.00m

```

You can also use selection together with processing-oriented commands.

```

# lvchange ---addtag test -S 'size < 5000k'
Logical volume vg/lvol1 changed.
Logical volume vg/lvol0 changed.
Logical volume vg/lvol3 changed.
Logical volume vg/lvol4 changed.

# lvchange ---deltag test -S 'tags = test'
Logical volume vg/lvol1 changed.
Logical volume vg/lvol0 changed.
Logical volume vg/lvol3 changed.
Logical volume vg/lvol4 changed.

```

LVM can recognize more complex values used in selection criteria for string list and time field types. For string lists, you can match whole list strictly, its subset or intersection. Let's take "lv_tags" field as an example – we select only rows which contain "tagA" within tags field. We're using { } to denote that we're interested in subset that matches. If the subset has only one item, we can leave out { }.

```

# lvs -o name,tags -S 'tags={tagA}'
LV  LV Tags
lvol2 tagA,tagC,tagD
lvol0 tagA,tagB

# lvs -o name,tags -S 'tags=tagA'
LV  LV Tags
lvol2 tagA,tagC,tagD
lvol0 tagA,tagB

```

Depending on whether we use "&&" (or ",") or "||" (or "#") as delimiter for items in the set we define in selection criterion for string list, we either match subset ("&&" or ",") or even intersection ("||" or "#").

```

# lvs -o name,tags -S 'tags={tagA,tagC,tagD}'
LV  LV Tags

```

```

lvol2 tagA,tagC,tagD

# lvs -o name,tags -S 'tags={tagA || tagC || tagD}'
LV  LV Tags
lvol2 tagA,tagC,tagD
lvol0 tagA,tagB

```

To match the complete set, use [] with "&&&" (or ",") as delimiter for items. Also note that the order in which we define items in the set is not relevant.

```

# lvs -o name,tags -S 'tags=[tagA]'

# lvs -o name,tags -S 'tags=[tagB,tagA]'
LV  LV Tags
lvol0 tagA,tagB

```

If you use [] with "||" (or "#"), this is exactly the same as using { }.

```

# lvs -o name,tags -S 'tags=[tagA || tagC || tagD]'
LV  LV Tags
lvol2 tagA,tagC,tagD
lvol0 tagA,tagB

```

To match a set with no items, use "" to denote this (note that we have output compaction enabled so the "LV Tags" column is not displayed in the example below because it's blank and so it gets compacted).

```

# lvs -o name,tags -S 'tags=""'
LV
lvol4
lvol3
lvol1

# lvs -o name,tags -S 'tags!=""'
LV  LV Tags
lvol2 tagA,tagC,tagD
lvol0 tagA,tagB

```

When doing selection based on time fields, we can use either standard, absolute or freeform time expressions in selection criteria. Examples below are using standard forms.

```

# lvs -o name,time
LV  CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time -S 'time since "2016-09-01"'
LV  CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200

```

```
# lvs -o name,time -S 'time since "2016-09-09 16:56"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200

# lvs -o name,time -S 'time since "2016-09-09 16:57:30"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200

# lvs -o name,time \
-S 'time since "2016-08-29" && time until "2016-09-09 16:55:12"'
LV CTime
lvol2 2016-09-09 16:55:12 +0200
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time \
-S 'time since "2016-08-29" && time before "2016-09-09 16:55:12"'
LV CTime
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200
```

Time operators have synonyms: ">=" for since, "<=" for until, ">" for "after" and "<" for "before".

```
# lvs -o name,time \
-S 'time >= "2016-08-29" && time <= "2016-09-09 16:55:30"'
LV CTime
lvol2 2016-09-09 16:55:12 +0200
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time \
-S 'time since "2016-08-29" && time < "2016-09-09 16:55:12"'
LV CTime
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200
```

Example below demonstrates using absolute time expression.

```
# lvs -o name,time --config report/time_format="%s"
LV CTime
lvol4 1473433064
lvol3 1473433008
lvol2 1473432912
lvol1 1472468016
lvol0 1472458517

# lvs -o name,time -S 'time since @1473433008'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
```

Examples below demonstrates using freeform time expressions.

```
# lvs -o name,time -S 'time since "2 weeks ago"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time -S 'time since "1 week ago"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200

# lvs -o name,time -S 'time since "2 weeks ago"'
LV CTime
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time -S 'time before "1 week ago"'
LV CTime
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200

# lvs -o name,time -S 'time since "68 hours ago"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200

# lvs -o name,time -S 'time since "1 year 3 months ago"'
LV CTime
lvol4 2016-09-09 16:57:44 +0200
lvol3 2016-09-09 16:56:48 +0200
lvol2 2016-09-09 16:55:12 +0200
lvol1 2016-08-29 12:53:36 +0200
lvol0 2016-08-29 10:15:17 +0200
```

Command log reporting

As described in **categorization based on reporting facility** section at the beginning of this document, both **report-oriented** and **processing-oriented** LVM commands can report the command log if this is enabled with **log/report_command_log** configuration setting. Just like any other report, we can set the set of fields to display (**log/command_log_cols**) and to sort by (**log/command_log_sort**) for this report.

```
# lvmconfig --type full log/report_command_log log/command_log_cols \
    log/command_log_sort log/command_log_selection
report_command_log=1
command_log_cols="log_seq_num,log_type,log_context,log_object_type,
    log_object_name,log_object_group,log_message,
    log_errno,log_ret_code"
command_log_sort="log_seq_num"
command_log_selection="!(log_type=status && message=success)"
```

```
# lvs
Logical Volume
=====
LV  LSize Cpy%Sync
lvol1 4.00m 100.00
lvol0 4.00m

Command Log
=====
Seq LogType Context ObjType ObjName ObjGrp Msg  Errno RetCode
```

As you can see, the command log is empty (it contains only field names). By default, LVM uses selection on the command log report and this case no row matched the selection criteria, see also **log report specifics** section in this document for more information. We're displaying complete log report in the example below where we can see that both LVs lvol0 and lvol1 were successfully processed as well as the VG vg they are part of.

```
# lvmconfig --type full log/command_log_selection
command_log_selection="all"

# lvs
Logical Volume
=====
LV  LSize Cpy%Sync
lvol1 4.00m 100.00
lvol0 4.00m

Command Log
=====
Seq LogType Context  ObjType ObjName ObjGrp Msg  Errno RetCode
 1 status processing lv    lvol0  vg    success  0   1
 2 status processing lv    lvol1  vg    success  0   1
 3 status processing vg    vg      success  0   1

# lvchange -an vg/lvol1
Command Log
=====
Seq LogType Context  ObjType ObjName ObjGrp Msg  Errno RetCode
 1 status processing lv    lvol1  vg    success  0   1
 2 status processing vg    vg      success  0   1
```

Handling multiple reports per single command

To configure the log report directly on command line, we need to use **--configreport** option before we start any **-o--options**, **-O--sort** or **-S--select** that is targeted for log report.

```
# lvs -o lv_name,lv_size --configreport log -o log_object_type, \
  log_object_name,log_message,log_ret_code
Logical Volume
=====
LV  LSize
lvol1 4.00m
lvol0 4.00m

Command Log
```

```
=====
ObjType ObjName Msg  RetCode
lv    lvol0  success  1
lv    lvol1  success  1
vg    vg     success  1
```

The **lvm fullreport**, with or without log report, consists of several reports – the **--configreport** is also used to target particular subreport here.

Below is an extended example with **lvm fullreport** to illustrate combination of various options. The report output is in JSON format. Also, we configure "vg", "pvseg", "seg" and "log" subreport to contain only specified fields. For the "pvseg" subreport, we're interested only in PV names having "sda" in their name. For the "log" subreport we're interested only in log lines related to either "lvol0" object or object having "sda" in its name. Also, for the log subreport we define ordering to be based on "log_object_type" field.

```
# lvm fullreport --reportformat json \
--configreport vg -o vg_name,vg_size \
--configreport pvseg -o pv_name,pvseg_start \
-S 'pv_name=~sda' \
--configreport seg -o lv_name,seg_start \
--configreport log -o log_object_type,log_object_name \
-O log_object_type \
-S 'log_object_name=lvol0 || \
log_object_name=~sda'
{
  "report": [
    {
      "vg": [
        {"vg_name": "vg", "vg_size": "200.00m"}
      ],
      "pv": [
        {"pv_name": "/dev/sda", "vg_name": "vg"},
        {"pv_name": "/dev/sdb", "vg_name": "vg"}
      ],
      "lv": [
        {"lv_name": "lvol0", "vg_name": "vg"},
        {"lv_name": "lvol1", "vg_name": "vg"}
      ],
      "pvseg": [
        {"pv_name": "/dev/sda", "pvseg_start": "0"}, 
        {"pv_name": "/dev/sda", "pvseg_start": "1"}, 
        {"pv_name": "/dev/sda", "pvseg_start": "2"}, 
        {"pv_name": "/dev/sda", "pvseg_start": "3"}
      ],
      "seg": [
        {"lv_name": "lvol0", "seg_start": "0 "}, 
        {"lv_name": "lvol1", "seg_start": "0 "}
      ]
    }
  ]
}
```

```

    ,
    "log": [
        {"log_object_type":"lv", "log_object_name":"lvol0"},  

        {"log_object_type":"lv", "log_object_name":"lvol0"},  

        {"log_object_type":"pv", "log_object_name":"/dev/sda"},  

        {"log_object_type":"pv", "log_object_name":"/dev/sda"},  

    ]
}

```

Report extensions for LVM shell

As already stated in **log report coverage** paragraph under **log report specifics** in this documentation, when using **LVM shell** the **log report** coverage is wider. There's also special command designed to query last command's log report in the **LVM shell** - the **lastlog** command.

The example below illustrates a situation where we called lvs command. After that, we inspected the log report with the **lastlog**, without any selection so all the log report is displayed on output. Then we called **lastlog** further, giving various selection criteria. Then we ran unknown LVM command "abc" for which the log report displays appropriate failure state.

```

# lvm
lvm> lvs
Logical Volume
=====
LV  LSize Cpy%Sync
lvol1 4.00m 100.00
lvol0 4.00m

Command Log
=====
Seq LogType Context ObjType ObjName ObjGrp Msg  Errno RetCode
1 status processing lv   lvol0  vg    success  0   1
2 status processing lv   lvol1  vg    success  0   1
3 status processing vg   vg      success  0   1
4 status shell   cmd   lvs     success  0   1

lvm> lastlog
Command Log
=====
Seq LogType Context ObjType ObjName ObjGrp Msg  Errno RetCode
1 status processing lv   lvol0  vg    success  0   1
2 status processing lv   lvol1  vg    success  0   1
3 status processing vg   vg      success  0   1
4 status shell   cmd   lvs     success  0   1

lvm> lastlog -S log_object_type=lv
Command Log
=====
Seq LogType Context ObjType ObjName ObjGrp Msg  Errno RetCode
1 status processing lv   lvol0  vg    success  0   1
2 status processing lv   lvol1  vg    success  0   1

lvm> lastlog -S log_context=shell
Command Log
=====
```

```
Seq LogType Context ObjType ObjName ObjGrp Msg    Errno RetCode
 4 status shell cmd   lvs      success  0     1

lvm> abc
Command Log
=====
Seq LogType Context ObjType ObjName ObjGrp Msg    Errno RetCode
 1 error shell cmd   abc      No such command 'abc'. Try 'help'. -1     0
 2 status shell cmd   abc      failure          -1     2
```

SEE ALSO

lvm (8), **lvmconfig** (8), **lvm fullreport** (8)

NAME

lvreduce – Reduce the size of a logical volume

SYNOPSIS

```
lvreduce option_args position_args
      [ option_args ]
```

DESCRIPTION

lvreduce reduces the size of an LV. The freed logical extents are returned to the VG to be used by other LVs. A copy-on-write snapshot LV can also be reduced if less space is needed to hold COW blocks. Use **lvconvert(8)** to change the number of data images in a RAID or mirrored LV.

Be careful when reducing an LV's size, because data in the reduced area is lost. Ensure that any file system on the LV is resized **before** running **lvreduce** so that the removed extents are not in use by the file system.

Sizes will be rounded if necessary. For example, the LV size must be an exact number of extents, and the size of a striped segment must be a multiple of the number of stripes.

In the usage section below, **--size** *Size* can be replaced with **--extents** *Number*. See both descriptions the options section.

USAGE

```
lvreduce -L|--size [ -Size[m|UNIT] ] LV
      [ -l|--extents [ -Number[PERCENT] ] ]
      [ -A|--autobackup y|n ]
      [ -f|--force ]
      [ -n|--nofsck ]
      [ -r|--resizefs ]
      [ --noudevsync ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-A|--autobackup *y|n*

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-l|--extents [-]Number[PERCENT]

Specifies the new size of the LV in logical extents. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. An alternate syntax allows the size to be determined indirectly as a percentage of the size of a related VG, LV, or set of PVs. The suffix **%VG** denotes the total size of the VG, the suffix **%FREE** the remaining free space in the VG, and the suffix **%PVS** the free space in the specified PVs. For a snapshot, the size can be expressed as a percentage of the total size of the origin LV with the suffix **%ORIGIN (100%ORIGIN** provides space for the whole origin). When expressed as a percentage, the size defines an upper limit for the number of logical extents in the new LV. The precise number of logical extents in the new LV is not determined until the command has completed. When the plus + or minus – prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-n|--nofsck

Do not perform fsck before resizing filesystem when filesystem requires it. You may need to use --force to proceed with this option.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-r|--resizesfs

Resize underlying filesystem together with the LV using fsadm(8).

-L|--size [-]Size[m|UNIT]

Specifies the new size of the LV. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. When the plus+ or minus – prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*LV*

Logical Volume name. See lvm(8) for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See lvm(8) for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Reduce the size of an LV by 3 logical extents:

lvreduce -l -3 vg00/lvol1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvremove – Remove logical volume(s) from the system

SYNOPSIS

```
lvremove position_args
      [ option_args ]
```

DESCRIPTION

lvremove removes one or more LVs. For standard LVs, this returns the logical extents that were used by the LV to the VG for use by other LVs.

Confirmation will be requested before deactivating any active LV prior to removal. LVs cannot be deactivated or removed while they are open (e.g. if they contain a mounted filesystem). Removing an origin LV will also remove all dependent snapshots.

When a single force option is used, LVs are removed without confirmation, and the command will try to deactivate unused LVs.

To remove damaged LVs, two force options may be required (**-ff**).

Historical LVs

If the configuration setting **metadata/record_lvs_history** is enabled and the LV being removed forms part of the history of at least one LV that is still present, then a simplified representation of the LV will be retained. This includes the time of removal (**lv_time_removed** reporting field), creation time (**lv_time**), name (**lv_name**), LV uuid (**lv_uuid**) and VG name (**vg_name**). This allows later reporting to see the ancestry chain of thin snapshot volumes, even after some intermediate LVs have been removed. The names of such historical LVs acquire a hyphen as a prefix (e.g. '-lvol1') and cannot be reactivated. Use **lvremove** a second time, with the hyphen, to remove the record of the former LV completely.

USAGE

```
lvremove VG|LV|Tag|Select ...
      [ -A|--autobackup y|n ]
      [ -f|--force ]
      [ -S|--select String ]
      [ --nohistory ]
      [ --noudevsync ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**-A|--autobackup y|n**

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vfcfgbackup(8)** for more information.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nohistory

Do not record history of LVs being removed. This has no effect unless the configuration setting metadata/record_lvs_history is enabled.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by **--select help** and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See **--options help** for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the

selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Remove an active LV without asking for confirmation.

lvremove -f vg00/lvol1

Remove all LVs the specified VG.

lvremove vg00

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisk(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvrename – Rename a logical volume

SYNOPSIS

```
lvrename position_args
        [ option_args ]
```

DESCRIPTION

lvrename renames an existing LV or a historical LV (see **lvremove** for historical LV information.)

USAGE

```
lvrename VG LV LV_new
        [ COMMON_OPTIONS ]
```

```
lvrename LV LV_new
        [ COMMON_OPTIONS ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ --noudevsync ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vfcfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGg-TtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is

TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Rename "lvold" to "lvnew":

```
lvrename /dev/vg02/lvold vg02/lvnew
```

An alternate syntax to rename "lvold" to "lvnew":

```
lvrename vg02 lvold lvnew
```

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvresize – Resize a logical volume

SYNOPSIS

```
lvresize option_args position_args
[ option_args ]
[ position_args ]
--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
-A|--autobackup y|n
--commandprofile String
--config String
-d|--debug
--driverloaded y|n
-l|--extents [+|-]Number[PERCENT]
-f|--force
-h|--help
--lockopt String
--longhelp
-n|--nofsck
--nolocking
--nosync
--noudevsync
--poolmetadatasize [+Size[m|UNIT]]
--profile String
-q|--quiet
--reportformat basic|json
-r|--resizesfs
-L|--size [+|-]Size[m|UNIT]
-i|--stripes Number
-I|--stripesize Size[k|UNIT]
-t|--test
--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool
-v|--verbose
--version
-y|--yes
```

DESCRIPTION

lvresize resizes an LV in the same way as **lvextend** and **lvreduce**. See **lvextend(8)** and **lvreduce(8)** for more information.

In the usage section below, **--size** *Size* can be replaced with **--extents** *Number*. See both descriptions the options section.

USAGE

Resize an LV by a specified size.

```
lvresize -L|--size [+|-]Size[m|UNIT] LV
[ -l|--extents [+|-]Number[PERCENT] ]
[ -r|--resizesfs ]
[ --poolmetadatasize [+Size[m|UNIT] ] ]
[ COMMON_OPTIONS ]
[ PV ... ]
```

Resize an LV by specified PV extents.

```
lvresize LV PV ...
```

```
[ -r|--resizesfs ]
[ COMMON_OPTIONS ]
```

Resize a pool metadata SubLV by a specified size.

```
lvresize --poolmetadatasize [+]Size[m|UNIT] LV_thinpool
[ COMMON_OPTIONS ]
[ PV ... ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ -n|--nofsck ]
[ -i|--stripes Number ]
[ -I|--stripesize Size[k|UNIT] ]
[ --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit ]
[ --nosync ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ --type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information

about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-l|--extents [+|-]Number[PERCENT]

Specifies the new size of the LV in logical extents. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. An alternate syntax allows the size to be determined indirectly as a percentage of the size of a related VG, LV, or set of PVs. The suffix **%VG** denotes the total size of the VG, the suffix **%FREE** the remaining free space in the VG, and the suffix **%PVS** the free space in the specified PVs. For a snapshot, the size can be expressed as a percentage of the total size of the origin LV with the suffix **%ORIGIN (100%ORIGIN** provides space for the whole origin). When expressed as a percentage, the size defines an upper limit for the number of logical extents in the new LV. The precise number of logical extents in the new LV is not determined until the command has completed. When the plus + or minus – prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-n|--nofsck

Do not perform fsck before resizing filesystem when filesystem requires it. You may need to use --force to proceed with this option.

--nolocking

Disable locking.

--nosync

Causes the creation of mirror, raid1, raid4, raid5 and raid10 to skip the initial synchronization. In case of mirror, raid1 and raid10, any data written afterwards will be mirrored, but the original contents will not be copied. In case of raid4 and raid5, no parity blocks will be written, though any data written afterwards will cause parity blocks to be stored. This is useful for skipping a potentially long and resource intensive initial sync of an empty mirror/raid1/raid4/raid5 and raid10 LV. This option is not valid for raid6, because raid6 relies on proper parity (P and Q Syndromes) being created during initial synchronization in order to reconstruct proper user data in case of device failures. raid0 and raid0_meta do not provide any data copies or parity support and thus do not support initial synchronization.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--poolmetadatasize [+]*Size[m|UNIT]*

Specifies the new size of the pool metadata LV. The plus prefix+ can be used, in which case the value is added to the current size.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-r|--resizesfs

Resize underlying filesystem together with the LV using fsadm(8).

-L|--size [+|-]*Size[m|UNIT]*

Specifies the new size of the LV. The --size and --extents options are alternate methods of specifying size. The total number of physical extents used will be greater when redundant data is needed for RAID levels. When the plus+ or minus – prefix is used, the value is not an absolute size, but is relative and added or subtracted from the current size.

-i|--stripes *Number*

Specifies the number of stripes in a striped LV. This is the number of PVs (devices) that a striped LV is spread across. Data that appears sequential in the LV is spread across multiple devices in units of the stripe size (see --stripesize). This does not change existing allocated space, but only applies to space being allocated by the command. When creating a RAID 4/5/6 LV, this number does not include the extra devices that are required for parity. The largest number depends on the RAID type (raid0: 64, raid10: 32, raid4/5: 63, raid6: 62), and when unspecified, the default depends on the RAID type (raid0: 2, raid10: 2, raid4/5: 3, raid6: 5.) To stripe a new raid LV across all PVs by default, see lvm.conf allocation/raid_stripe_all_devices.

-I|--stripesize *Size[k|UNIT]*

The amount of data that is written to one device before moving to the next in a striped LV.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--type linear|striped|snapshot|mirror|raid|thin|cache|vdo|thin-pool|cache-pool|vdo-pool

The LV type, also known as "segment type" or "segtype". See usage descriptions for the specific ways to use these types. For more information about redundancy and performance (**raid<N>**, **mirror**, **striped**, **linear**) see **lvmraid(7)**. For thin provisioning (**thin**, **thin-pool**) see **lvmthin(7)**. For performance caching (**cache**, **cache-pool**) see **lvmcache(7)**. For copy-on-write snapshots (**snapshot**) see usage definitions. For VDO (**vdo**) see **lvmvdo(7)**. Several commands omit an explicit type option because the type is inferred from other options or shortcuts (e.g. --stripes, --mirrors, --snapshot, --virtualsize, --thin, --cache, --vdo). Use inferred types with care because it can lead to unexpected results.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **--qq**.)

VARIABLES*LV*

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV. LV followed by _<type> indicates that an LV of the given type is required. (raid represents raid<N> type)

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Extend an LV by 16MB using specific physical extents:
lvresize -L+16M vg1/lv1 /dev/sda:0-1 /dev/sdb:0-1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvs – Display information about logical volumes

SYNOPSIS

```
lvs
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

lvs produces formatted output about LVs.

USAGE

```
lvs
  [ -H|--history ]
  [ -a|--all ]
  [ -o|--options String ]
  [ -S|--select String ]
  [ -O|--sort String ]
  [ --segments ]
  [ --aligned ]
  [ --binary ]
  [ --configreport log|vg|lv|pv|pvseg|seg ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --logonly ]
  [ --nameprefixes ]
  [ --noheadings ]
  [ --nosuffix ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ --rows ]
  [ --separator String ]
  [ --shared ]
  [ --unbuffered ]
  [ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
  [ --unquoted ]
  [ COMMON_OPTIONS ]
  [ VG|LV|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--aligned

Use with --separator to align the output columns

-a|--all

Show information about internal LVs. These are components of normal LVs, such as mirrors, which are not independently accessible, e.g. not mountable.

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport log|vg|lv|pv|pvseg|seg

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

-H|--history

Include historical LVs in the output. (This has no effect unless LVs were removed while lvm.conf metadata/record_lvs_history was enabled.)

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

--nameprefixes

Add an "LVM2_" prefix plus the field name to the output. Useful with --noheadings to produce a list of field=value pairs that can be used to set environment variables (for example, in udev rules).

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with **--units** (except h and H) if processing the output.

-o|--options String

Comma-separated, ordered list of fields to display in columns. String arg syntax is:
`[+|-#]Field1[,Field2 ...]` The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the **-o** option: `-o+field1,field2 -o- field3,field4 -o#field5`. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **Ivmreport(7)** for more information about reporting.

--profile String

An alias for **--commandprofile** or **--metadataprofile**, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides **--debug** and **--verbose**. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **Ivmreport(7)** for more information.

--rows

Output columns as rows.

--segments

Use default columns that emphasize segment information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by **--select help** and **Ivmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See **--options help** for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator String

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when lvmlockd is not being used on the host. See **Ivmlockd(8)** for more information about shared VGs.

-O|--sort String

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any column with - for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes

has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-read-able, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes.

Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g.
--units 3M.

--unquoted

When used with --nameprefixes, output values in the field=value pairs are not quoted.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

The lv_attr bits are:

- 1 Volume type: (C)ache, (m)irrored, (M)irrored without initial sync, (o)rigin, (O)rigin with merging snapshot, (r)aid, (R)aid without initial sync, (s)napshot, merging (S)napshot, (p)vmove, (v)irtual, mirror or raid (i)mage, mirror or raid (I)mage out-of-sync, mirror (l)og device, under (c)onversion, thin (V)olume, (t)hin pool, (T)hin pool data, v(d)o pool, v(D)o pool data, raid or pool m(e)tadata or pool metadata spare.
- 2 Permissions: (w)riteable, (r)ead-only, (R)ead-only activation of non-read-only volume
- 3 Allocation policy: (a)nywhere, (c)ontiguous, (i)nherited, c(l)ing, (n)ormal This is capitalised if the volume is currently locked against allocation changes, for example during **pvmove(8)**.

- 4 fixed (m)inor
- 5 State: (a)ctive, (h)istorical, (s)suspended, (I)nvalid snapshot, invalid (S)suspended snapshot, snapshot (m)erge failed, suspended snapshot (M)erge failed, mapped (d)evice present without tables, mapped device present with (i)nactive table, thin-pool (c)heck needed, suspended thin-pool (C)heck needed, (X) unknown
- 6 device (o)pen, (X) unknown
- 7 Target type: (C)ache, (m)irror, (r)aid, (s)napshot, (t)hin, (u)nknown, (v)irtual. This groups logical volumes related to the same kernel target together. So, for example, mirror images, mirror logs as well as mirrors themselves appear as (m) if they use the original device-mapper mirror kernel driver; whereas the raid equivalents using the md raid kernel driver all appear as (r). Snapshots using the original device-mapper driver appear as (s); whereas snapshots of thin volumes using the new thin provisioning driver appear as (t).
- 8 Newly-allocated data blocks are overwritten with blocks of (z)eroes before use.
- 9 Volume Health, where there are currently three groups of attributes identified:
 - Common ones for all Logical Volumes: (p)artial, (X) unknown.
 - (p)artial signifies that one or more of the Physical Volumes this Logical Volume uses is missing from the system. (X) unknown signifies the status is unknown.
 - Related to RAID Logical Volumes: (r)efresh needed, (m)ismatches exist, (w)ritemostly.
 - (r)efresh signifies that one or more of the Physical Volumes this RAID Logical Volume uses had suffered a write error. The write error could be due to a temporary failure of that Physical Volume or an indication that it is failing. The device should be refreshed or replaced.
 - (m)ismatches signifies that the RAID logical volume has portions of the array that are not coherent. Inconsistencies are detected by initiating a "check" on a RAID logical volume. (The scrubbing operations, "check" and "repair", can be performed on a RAID logical volume via the 'lvchange' command.)
 - (w)ritemostly signifies the devices in a RAID 1 logical volume that have been marked write-mostly. Re(s)haping signifies a RAID Logical Volume is either undergoing a stripe addition/removal, a stripe size or RAID algorithm change.
 - (R)emove after reshape signifies freed striped raid images to be removed.
 - Related to Thin pool Logical Volumes: (F)ailed, out of (D)ata space, (M)etadata read only.
 - (F)ailed is set if thin pool encounters serious failures and hence no further I/O is permitted at all.
 - The out of (D)ata space is set if thin pool has run out of data space.
 - (M)etadata read only signifies that thin pool encounters certain types of failures but it's still possible to do reads at least, but no metadata changes are allowed.
 - Related to Thin Logical Volumes: (F)ailed.
 - (F)ailed is set when related thin pool enters Failed state and no further I/O is permitted at all.
 - Related to writecache logical volumes: (E)rror.
 - (E)rror is set dm-writecache reports an error.
- 10 s(k)ip activation: this volume is flagged to be skipped during activation.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisk(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

lvscan – List all logical volumes in all volume groups

SYNOPSIS

lvscan

[*option_args*]

DESCRIPTION

lvscan scans all VGs or all supported LVM block devices in the system for LVs. The output consists of one line for each LV indicating whether or not it is active, a snapshot or origin, the size of the device and its allocation policy. Use **lvs(8)** or **lvdisplay(8)** to obtain more comprehensive information about LVs.

USAGE

lvscan

[**-a|--all**]
[**-b|--blockdevice**]
[**--ignorelockingfailure**]
[**--readonly**]
[**--reportformat basic|json**]
[COMMON_OPTIONS]

Common options for lvm:

[**-d|--debug**]
[**-h|--help**]
[**-q|--quiet**]
[**-t|--test**]
[**-v|--verbose**]
[**-y|--yes**]
[**--commandprofile** *String*]
[**--config** *String*]
[**--driverloaded** **y|n**]
[**--lockopt** *String*]
[**--longhelp**]
[**--nolocking**]
[**--profile** *String*]
[**--version**]

OPTIONS

-a|--all

Show information about internal LVs. These are components of normal LVs, such as mirrors, which are not independently accessible, e.g. not mountable.

-b|--blockdevice

No longer used.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded **y|n**

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*String*

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

man – an interface to the system reference manuals

SYNOPSIS

```
man [ man options] [ [section] pa ge ...] ...
man -k [ apropos options] regexp ...
man -K [ man options] [section] term ...
man -f [ whatis options] pa ge ...
man -l [ man options] file ...
man -w|-W [ man options] pa ge ...
```

DESCRIPTION

man is the system's manual pager. Each *pa ge* argument given to **man** is normally the name of a program, utility or function. The *manual page* associated with each of these arguments is then found and displayed. A *section*, if provided, will direct **man** to look only in that *section* of the manual. The default action is to search in all of the available *sections* following a pre-defined order (see **DEFAULTS**), and to show only the first *page* found, even if *page* exists in several *sections*.

The table below shows the *section* numbers of the manual followed by the types of pages they contain.

1	Executable programs or shell commands
2	System calls (functions provided by the kernel)
3	Library calls (functions within program libraries)
4	Special files (usually found in <i>/dev</i>)
5	File formats and conventions, e.g. <i>/etc/passwd</i>
6	Games
7	Miscellaneous (including macro packages and conventions), e.g. man (7), groff (7), man-pages (7)
8	System administration commands (usually only for root)
9	Kernel routines [Non standard]

A manual *page* consists of several sections.

Conventional section names include **NAME**, **SYNOPSIS**, **CONFIGURATION**, **DESCRIPTION**, **OPTIONS**, **EXIT STATUS**, **RETURN VALUE**, **ERRORS**, **ENVIRONMENT**, **FILES**, **VERSIONS**, **CONFORMING TO**, **NOTES**, **BUGS**, **EXAMPLE**, **AUTHORS**, and **SEE ALSO**.

The following conventions apply to the **SYNOPSIS** section and can be used as a guide in other sections.

bold text	type exactly as shown.
<i>italic text</i>	replace with appropriate argument.
[-abc]	any or all arguments within [] are optional.
-a -b	options delimited by cannot be used together.
<i>argument</i> ...	<i>argument</i> is repeatable.
[<i>expression</i>] ...	entire <i>expression</i> within [] is repeatable.

Exact rendering may vary depending on the output device. For instance, man will usually not be able to render italics when running in a terminal, and will typically use underlined or coloured text instead.

The command or function illustration is a pattern that should match all possible invocations. In some cases it is advisable to illustrate several exclusive invocations as is shown in the **SYNOPSIS** section of this manual page.

EXAMPLES

man ls

Display the manual page for the *item* (program) *ls*.

man man.7

Display the manual page for macro package *man* from section 7. (This is an alternative spelling of "**man 7 man**".)

man 'man(7)'

Display the manual page for macro package *man* from section 7. (This is another alternative spelling of "**man 7 man**". It may be more convenient when copying and pasting cross-references to manual pages. Note that the parentheses must normally be quoted to protect them from the shell.)

man -a intro

Display, in succession, all of the available *intro* manual pages contained within the manual. It is possible to quit between successive displays or skip any of them.

man -t bash | lpr -Pps

Format the manual page for *bash* into the default **troff** or **groff** format and pipe it to the printer named *ps*. The default output for **groff** is usually PostScript. **man --help** should advise as to which processor is bound to the **-t** option.

man -l -Tdvi ./foo.1x.gz > ./foo.1x.dvi

This command will decompress and format the nroff source manual page *./foo.1x.gz* into a **device independent (dvi)** file. The redirection is necessary as the **-T** flag causes output to be directed to **stdout** with no pager. The output could be viewed with a program such as **xdvi** or further processed into PostScript using a program such as **dvips**.

man -k printf

Search the short descriptions and manual page names for the keyword *printf* as regular expression. Print out any matches. Equivalent to **apropos printf**.

man -f smail

Lookup the manual pages referenced by *smail* and print out the short descriptions of any found. Equivalent to **whatis smail**.

OVERVIEW

Many options are available to **man** in order to give as much flexibility as possible to the user. Changes can be made to the search path, section order, output processor, and other behaviours and operations detailed below.

If set, various environment variables are interrogated to determine the operation of **man**. It is possible to set the "catch-all" variable **\$MANOPT** to any string in command line format, with the exception that any spaces used as part of an option's argument must be escaped (preceded by a backslash). **man** will parse **\$MANOPT** prior to parsing its own command line. Those options requiring an argument will be overridden by the same options found on the command line. To reset all of the options set in **\$MANOPT**, **-D** can be specified as the initial command line option. This will allow man to "forget" about the options specified in **\$MANOPT**, although they must still have been valid.

Manual pages are normally stored in **nroff(1)** format under a directory such as */usr/share/man*. In some installations, there may also be preformatted *cat pages* to improve performance. See **manpath(5)** for details of where these files are stored.

This package supports manual pages in multiple languages, controlled by your *locale*. If your system did not set this up for you automatically, then you may need to set **\$LC_MESSAGES**, **\$LANG**, or another system-dependent environment variable to indicate your preferred locale, usually specified in the **POSIX** format:

<language>[_<territory>[.<character-set>[,<version>]]]

If the desired page is available in your *locale*, it will be displayed in lieu of the standard (usually American English) page.

If you find that the translations supplied with this package are not available in your native language and you would like to supply them, please contact the maintainer who will be coordinating such activity.

Individual manual pages are normally written and maintained by the maintainers of the program, function, or other topic that they document, and are not included with this package. If you find that a manual page is missing or inadequate, please report that to the maintainers of the package in question.

For information regarding other features and extensions available with this manual pager, please read the documents supplied with the package.

DEFUALTS

The order of sections to search may be overridden by the environment variable \$MANSECT or by the **SECTION** directive in */etc/manpath.config*. By default it is as follows:

```
1 n l 8 3 0 2 3posix 3pm 3perl 3am 5 4 9 6 7
```

The formatted manual page is displayed using a *pager*. This can be specified in a number of ways, or else will fall back to a default (see option **-P** for details).

The filters are deciphered by a number of means. Firstly, the command line option **-p** or the environment variable \$MANOFFSEQ is interrogated. If **-p** was not used and the environment variable was not set, the initial line of the nroff file is parsed for a preprocessor string. To contain a valid preprocessor string, the first line must resemble

```
'\" <string>
```

where **string** can be any combination of letters described by option **-p** below.

If none of the above methods provide any filter information, a default set is used.

A formatting pipeline is formed from the filters and the primary formatter (**nroff** or **[tg]roff** with **-t**) and executed. Alternatively, if an executable program *mandb_nfnt* (or *mandb_tfmt* with **-t**) exists in the man tree root, it is executed instead. It gets passed the manual source file, the preprocessor string, and optionally the device specified with **-T** or **-E** as arguments.

OPTIONS

Non-argument options that are duplicated either on the command line, in \$MANOPT, or both, are not harmful. For options that require an argument, each duplication will override the previous argument value.

General options

-C *file*, **--config-file**=*file*

Use this user configuration file rather than the default of *~/.manpath*.

-d, **--debug**

Print debugging information.

-D, **--default**

This option is normally issued as the very first option and resets **man**'s behaviour to its default. Its use is to reset those options that may have been set in \$MANOPT. Any options that follow **-D** will have their usual effect.

--warnings[=*warnings*]

Enable warnings from *groff*. This may be used to perform sanity checks on the source text of manual pages. *warnings* is a comma-separated list of warning names; if it is not supplied, the default is "mac". See the "Warnings" node in **inf o groff** for a list of available warning names.

Main modes of operation**-f, --whatis**

Equivalent to **whatis**. Display a short description from the manual page, if available. See **whatis(1)** for details.

-k, --apropos

Equivalent to **apropos**. Search the short manual page descriptions for keywords and display any matches. See **apropos(1)** for details.

-K, --global-apropos

Search for text in all manual pages. This is a brute-force search, and is likely to take some time; if you can, you should specify a section to reduce the number of pages that need to be searched. Search terms may be simple strings (the default), or regular expressions if the **--regex** option is used.

Note that this searches the *sources* of the manual pages, not the rendered text, and so may include false positives due to things like comments in source files. Searching the rendered text would be much slower.

-l, --local-file

Activate "local" mode. Format and display local manual files instead of searching through the system's manual collection. Each manual page argument will be interpreted as an nroff source file in the correct format. No cat file is produced. If '-' is listed as one of the arguments, input will be taken from stdin. When this option is not used, and man fails to find the page required, before displaying the error message, it attempts to act as if this option was supplied, using the name as a file-name and looking for an exact match.

-w, --where, --path, --location

Don't actually display the manual page, but do print the location of the source nroff file that would be formatted. If the **-a** option is also used, then print the locations of all source files that match the search criteria.

-W, --where-cat, --location-cat

Don't actually display the manual page, but do print the location of the preformatted cat file that would be displayed. If the **-a** option is also used, then print the locations of all preformatted cat files that match the search criteria.

If **-w** and **-W** are both used, then print both source file and cat file separated by a space. If all of **-w**, **-W**, and **-a** are used, then do this for each possible match.

-c, --catman

This option is not for general use and should only be used by the **catman** program.

-R encoding, --recode=encoding

Instead of formatting the manual page in the usual way, output its source converted to the specified *encoding*. If you already know the encoding of the source file, you can also use **manconv(1)** directly. However, this option allows you to convert several manual pages to a single encoding without having to explicitly state the encoding of each, provided that they were already installed in a structure similar to a manual page hierarchy.

Consider using **man-recode(1)** instead for converting multiple manual pages, since it has an interface designed for bulk conversion and so can be much faster.

Finding manual pages**-L locale, --locale=locale**

man will normally determine your current locale by a call to the C function **setlocale(3)** which interrogates various environment variables, possibly including **\$LC_MESSAGES** and **\$LANG**. To temporarily override the determined value, use this option to supply a *locale* string directly to **man**. Note that it will not take effect until the search for pages actually begins. Output such as the help message will always be displayed in the initially determined locale.

-m system [,...] , --systems=system [,...]

If this system has access to other operating systems' manual pages, they can be accessed using this option. To search for a manual page from NewOS's manual page collection, use the option **-m NewOS**.

The *system* specified can be a combination of comma delimited operating system names. To include a search of the native operating system's manual pages, include the system name **man** in the argument string. This option will override the **\$SYSTEM** environment variable.

-M path, --manpath=path

Specify an alternate manpath to use. By default, **man** uses **manpath** derived code to determine the path to search. This option overrides the **\$MANPATH** environment variable and causes option **-m** to be ignored.

A path specified as a manpath must be the root of a manual page hierarchy structured into sections as described in the man-db manual (under "The manual page system"). To view manual pages outside such hierarchies, see the **-l** option.

-S list, -s list, --sections=list

The given *list* is a colon- or comma-separated list of sections, used to determine which manual sections to search and in what order. This option overrides the **\$MANSECT** environment variable. (The **-s** spelling is for compatibility with System V.)

-e sub-extension, --extension=sub-extension

Some systems incorporate large packages of manual pages, such as those that accompany the **Tcl** package, into the main manual page hierarchy. To get around the problem of having two manual pages with the same name such as **exit(3)**, the **Tcl** pages were usually all assigned to section **I**. As this is unfortunate, it is now possible to put the pages in the correct section, and to assign a specific "extension" to them, in this case, **exit(3tcl)**. Under normal operation, **man** will display **exit(3)** in preference to **exit(3tcl)**. To negotiate this situation and to avoid having to know which section the page you require resides in, it is now possible to give **man** a *sub-extension* string indicating which package the page must belong to. Using the above example, supplying the option **-e tcl** to **man** will restrict the search to pages having an extension of ***tcl**.

-i, --ignore-case

Ignore case when searching for manual pages. This is the default.

-I, --match-case

Search for manual pages case-sensitively.

--regex

Show all pages with any part of either their names or their descriptions matching each *page* argument as a regular expression, as with **apropos(1)**. Since there is usually no reasonable way to pick a "best" page when searching for a regular expression, this option implies **-a**.

--wildcard

Show all pages with any part of either their names or their descriptions matching each *page* argument using shell-style wildcards, as with **apropos(1)** **--wildcard**. The *page* argument must match the entire name or description, or match on word boundaries in the description. Since there is usually no reasonable way to pick a "best" page when searching for a wildcard, this option implies **-a**.

--names-only

If the **--regex** or **--wildcard** option is used, match only page names, not page descriptions, as with **whatis(1)**. Otherwise, no effect.

-a, --all

By default, **man** will exit after displaying the most suitable manual page it finds. Using this option forces **man** to display all the manual pages with names that match the search criteria.

-u, --update

This option causes **man** to update its database caches of installed manual pages. This is only needed in rare situations, and it is normally better to run **mandb(8)** instead.

--no-subpages

By default, **man** will try to interpret pairs of manual page names given on the command line as equivalent to a single manual page name containing a hyphen or an underscore. This supports the common pattern of programs that implement a number of subcommands, allowing them to provide manual pages for each that can be accessed using similar syntax as would be used to invoke the subcommands themselves. For example:

```
$ man -aw git diff
/usr/share/man/man1/git-diff.1.gz
```

To disable this behaviour, use the **--no-subpages** option.

```
$ man -aw --no-subpages git diff
/usr/share/man/man1/git.1.gz
/usr/share/man/man3/Git.3pm.gz
/usr/share/man/man1/diff.1.gz
```

Controlling formatted output**-P pager, --pager=pager**

Specify which output pager to use. By default, **man** uses **pager**, falling back to **cat** if **pager** is not found or is not executable. This option overrides the **\$MANPAGER** environment variable, which in turn overrides the **\$PAGER** environment variable. It is not used in conjunction with **-f** or **-k**.

The value may be a simple command name or a command with arguments, and may use shell quoting (backslashes, single quotes, or double quotes). It may not use pipes to connect multiple commands; if you need that, use a wrapper script, which may take the file to display either as an argument or on standard input.

-r prompt, --prompt=prompt

If a recent version of **less** is used as the pager, **man** will attempt to set its prompt and some sensible options. The default prompt looks like

Manual page name(sec) line x

where *name* denotes the manual page name, *sec* denotes the section it was found under and *x* the current line number. This is achieved by using the **\$LESS** environment variable.

Supplying **-r** with a string will override this default. The string may contain the text **\$MAN_PN** which will be expanded to the name of the current manual page and its section name surrounded by "(" and ")". The string used to produce the default could be expressed as

```
\ Manual\ page\ $MAN_PN\ ?ltline\ %lt?L/%L.:
byte\ %bB?s/%s..?\ (END):?pB\ %pB\\%..
(press h for help or q to quit)
```

It is broken into three lines here for the sake of readability only. For its meaning see the **less(1)** manual page. The prompt string is first evaluated by the shell. All double quotes, back-quotes and backslashes in the prompt must be escaped by a preceding backslash. The prompt string may end in an escaped \$ which may be followed by further options for less. By default **man** sets the **-ix8** options.

The **\$MANLESS** environment variable described below may be used to set a default prompt

string if none is supplied on the command line.

-7, --ascii

When viewing a pure *ascii*(7) manual page on a 7 bit terminal or terminal emulator, some characters may not display correctly when using the *latin1*(7) device description with **GNU nroff**. This option allows pure *ascii* manual pages to be displayed in *ascii* with the *latin1* device. It will not translate any *latin1* text. The following table shows the translations performed: some parts of it may only be displayed properly when using **GNU nroff**'s *latin1*(7) device.

Description	Octal	latin1	ascii
continuation hyphen	255	-	-
bullet (middle dot)	267	•	o
acute accent	264	ˊ	,
multiplication sign	327	×	x

If the *latin1* column displays correctly, your terminal may be set up for *latin1* characters and this option is not necessary. If the *latin1* and *ascii* columns are identical, you are reading this page using this option or **man** did not format this page using the *latin1* device description. If the *latin1* column is missing or corrupt, you may need to view manual pages with this option.

This option is ignored when using options **-t**, **-H**, **-T**, or **-Z** and may be useless for **nroff** other than **GNU**'s.

-E encoding, --encoding=encoding

Generate output for a character encoding other than the default. For backward compatibility, *encoding* may be an **nroff** device such as **ascii**, **latin1**, or **utf8** as well as a true character encoding such as **UTF-8**.

--no-hyphenation, --nh

Normally, **nroff** will automatically hyphenate text at line breaks even in words that do not contain hyphens, if it is necessary to do so to lay out words on a line without excessive spacing. This option disables automatic hyphenation, so words will only be hyphenated if they already contain hyphens.

If you are writing a manual page and simply want to prevent **nroff** from hyphenating a word at an inappropriate point, do not use this option, but consult the **nroff** documentation instead; for instance, you can put "%\" inside a word to indicate that it may be hyphenated at that point, or put "%\" at the start of a word to prevent it from being hyphenated.

--no-justification, --nj

Normally, **nroff** will automatically justify text to both margins. This option disables full justification, leaving justified only to the left margin, sometimes called "ragged-right" text.

If you are writing a manual page and simply want to prevent **nroff** from justifying certain paragraphs, do not use this option, but consult the **nroff** documentation instead; for instance, you can use the ".na", ".nf", ".fi", and ".ad" requests to temporarily disable adjusting and filling.

-p string, --preprocessor=string

Specify the sequence of preprocessors to run before **nroff** or **troff/groff**. Not all installations will have a full set of preprocessors. Some of the preprocessors and the letters used to designate them are: **eqn** (**e**), **grap** (**g**), **pic** (**p**), **tbl** (**t**), **vgrind** (**v**), **refer** (**r**). This option overrides the **\$MAN-ROFFSEQ** environment variable. **zsoelim** is always run as the very first preprocessor.

-t, --troff

Use **groff -mandoc** to format the manual page to stdout. This option is not required in conjunction with **-H**, **-T**, or **-Z**.

-T[device], --troff-device[=device]

This option is used to change **groff** (or possibly **troff**'s) output to be suitable for a device other than the default. It implies **-t**. Examples (provided with Groff-1.17) include **dvi**, **latin1**, **ps**, **utf8**, **X75** and **X100**.

-H[browser], --html[=browser]

This option will cause **groff** to produce HTML output, and will display that output in a web browser. The choice of browser is determined by the optional *browser* argument if one is provided, by the **\$BROWSER** environment variable, or by a compile-time default if that is unset (usually **lynx**). This option implies **-t**, and will only work with GNU **troff**.

-X[dpi], --gxditview[=dpi]

This option displays the output of **groff** in a graphical window using the **gxditview** program. The *dpi* (dots per inch) may be 75, 75-12, 100, or 100-12, defaulting to 75; the -12 variants use a 12-point base font. This option implies **-T** with the X75, X75-12, X100, or X100-12 device respectively.

-Z, --ditroff

groff will run **troff** and then use an appropriate post-processor to produce output suitable for the chosen device. If *off -mandoc* is **groff**, this option is passed to **groff** and will suppress the use of a post-processor. It implies **-t**.

Getting help**-?, --help**

Print a help message and exit.

--usage

Print a short usage message and exit.

-V, --version

Display version information.

EXIT STATUS

- | | |
|-----------|--|
| 0 | Successful program execution. |
| 1 | Usage, syntax or configuration file error. |
| 2 | Operational error. |
| 3 | A child process returned a non-zero exit status. |
| 16 | At least one of the pages/files/keywords didn't exist or wasn't matched. |

ENVIRONMENT**MANPATH**

If **\$MANPATH** is set, its value is used as the path to search for manual pages.

See the **SEARCH PATH** section of **manpath(5)** for the default behaviour and details of how this environment variable is handled.

MANROFOPT

Every time **man** invokes the formatter (**nroff**, **troff**, or **groff**), it adds the contents of **\$MANROFOPT** to the formatter's command line.

MANOFFSEQ

If **\$MANOFFSEQ** is set, its value is used to determine the set of preprocessors to pass each manual page through. The default preprocessor list is system dependent.

MANSECT

If **\$MANSECT** is set, its value is a colon-delimited list of sections and it is used to determine which manual sections to search and in what order. The default is "1 n 1 8 3 0 2 3posix 3pm 3perl 3am 5 4 9 6 7", unless overridden by the **SECTION** directive in */etc/manpath.config*.

MANPAGER, PAGER

If **\$MANPAGER** or **\$PAGER** is set (**\$MANPAGER** is used in preference), its value is used as the name of the program used to display the manual page. By default, **pager** is used, falling back to **cat** if **pager** is not found or is not executable.

The value may be a simple command name or a command with arguments, and may use shell quoting (backslashes, single quotes, or double quotes). It may not use pipes to connect multiple commands; if you need that, use a wrapper script, which may take the file to display either as an argument or on standard input.

MANLESS

If **\$MANLESS** is set, its value will be used as the default prompt string for the **less** pager, as if it had been passed using the **-r** option (so any occurrences of the text **\$MAN_PN** will be expanded in the same way). For example, if you want to set the prompt string unconditionally to “my prompt string”, set **\$MANLESS** to ‘**-P***my prompt string*’. Using the **-r** option overrides this environment variable.

BROWSER

If **\$BROWSER** is set, its value is a colon-delimited list of commands, each of which in turn is used to try to start a web browser for **man --html**. In each command, **%s** is replaced by a file-name containing the HTML output from **groff**, **%%** is replaced by a single percent sign (%), and **%c** is replaced by a colon (:).

SYSTEM

If **\$SYSTEM** is set, it will have the same effect as if it had been specified as the argument to the **-m** option.

MANOPT

If **\$MANOPT** is set, it will be parsed prior to **man**’s command line and is expected to be in a similar format. As all of the other **man** specific environment variables can be expressed as command line options, and are thus candidates for being included in **\$MANOPT** it is expected that they will become obsolete. N.B. All spaces that should be interpreted as part of an option’s argument must be escaped.

MANWIDTH

If **\$MANWIDTH** is set, its value is used as the line length for which manual pages should be formatted. If it is not set, manual pages will be formatted with a line length appropriate to the current terminal (using the value of **\$COLUMNS**, and **ioctl(2)** if available, or falling back to 80 characters if neither is available). Cat pages will only be saved when the default formatting can be used, that is when the terminal line length is between 66 and 80 characters.

MAN_KEEP_FORMATTING

Normally, when output is not being directed to a terminal (such as to a file or a pipe), formatting characters are discarded to make it easier to read the result without special tools. However, if **\$MAN_KEEP_FORMATTING** is set to any non-empty value, these formatting characters are retained. This may be useful for wrappers around **man** that can interpret formatting characters.

MAN_KEEP_STDERR

Normally, when output is being directed to a terminal (usually to a pager), any error output from the command used to produce formatted versions of manual pages is discarded to avoid interfering with the pager’s display. Programs such as **gr off** often produce relatively minor error messages about typographical problems such as poor alignment, which are unsightly and generally confusing when displayed along with the manual page. However, some users want to see them anyway, so, if **\$MAN_KEEP_STDERR** is set to any non-empty value, error output will be displayed as usual.

MAN_DISABLE_SECCOMP

On Linux, **man** normally confines subprocesses that handle untrusted data using a **seccomp(2)** sandbox. This makes it safer to run complex parsing code over arbitrary manual pages. If this

goes wrong for some reason unrelated to the content of the page being displayed, you can set **\$MAN_DISABLE_SECCOMP** to any non-empty value to disable the sandbox.

PIPELINE_DEBUG

If the **\$PIPELINE_DEBUG** environment variable is set to "1", then **man** will print debugging messages to standard error describing each subprocess it runs.

LANG, LC_MESSAGES

Depending on system and implementation, either or both of **\$LANG** and **\$LC_MESSAGES** will be interrogated for the current message locale. **man** will display its messages in that locale (if available). See **setlocale(3)** for precise details.

FILES

/etc/manpath.config
man-db configuration file.

/usr/share/man
A global manual page hierarchy.

SEE ALSO

apropos(1), groff(1), less(1), manpath(1), nroff(1), troff(1), whatis(1), zsoelim(1), manpath(5), man(7), catman(8), mandb(8)

Documentation for some packages may be available in other formats, such as **info(1)** or **HTML**.

HISTORY

1990, 1991 – Originally written by John W. Eaton (jwe@che.utexas.edu).

Dec 23 1992: Rik Faith (faith@cs.unc.edu) applied bug fixes supplied by Willem Kasdorp (wkasdo@nikhefk.nikef.nl).

30th April 1994 – 23rd February 2000: Wilf. (G.Wilford@ee.surrey.ac.uk) has been developing and maintaining this package with the help of a few dedicated people.

30th October 1996 – 30th March 2001: Fabrizio Polacco <fpolacco@debian.org> maintained and enhanced this package for the Debian project, with the help of all the community.

31st March 2001 – present day: Colin Watson <cjwatson@debian.org> is now developing and maintaining man-db.

BUGS

<https://gitlab.com/cjwatson/man-db/-/issues>
<https://savannah.nongnu.org/bugs/?group=man-db>

NAME

man – macros to format man pages

SYNOPSIS

```
groff -Tascii -man file ...
groff -Tp -man file ...
man [section] title
```

DESCRIPTION

This manual page explains the **groff an.tmac** macro package (often called the **man** macro package). This macro package should be used by developers when writing or porting man pages for Linux. It is fairly compatible with other versions of this macro package, so porting man pages should not be a major problem (exceptions include the NET-2 BSD release, which uses a totally different macro package called mdoc; see **mdoc(7)**).

Note that NET-2 BSD mdoc man pages can be used with **groff** simply by specifying the **-mdoc** option instead of the **-man** option. Using the **-mandoc** option is, however, recommended, since this will automatically detect which macro package is in use.

For conventions that should be employed when writing man pages for the Linux *man-pages* package, see **man-pages(7)**.

Title line

The first command in a man page (after comment lines, that is, lines that start with .\") should be

```
.TH title section date source manual
```

For details of the arguments that should be supplied to the **TH** command, see **man-pages(7)**.

Note that BSD mdoc-formatted pages begin with the **Dd** command, not the **TH** command.

Sections

Sections are started with **.SH** followed by the heading name.

The only mandatory heading is NAME, which should be the first section and be followed on the next line by a one-line description of the program:

```
.SH NAME
item \- description
```

It is extremely important that this format is followed, and that there is a backslash before the single dash which follows the item name. This syntax is used by the **mandb(8)** program to create a database of short descriptions for the **whatis(1)** and **apropos(1)** commands. (See **lexgrog(1)** for further details on the syntax of the NAME section.)

For a list of other sections that might appear in a manual page, see **man-pages(7)**.

Fonts

The commands to select the type face are:

.B Bold

.BI Bold alternating with italics (especially useful for function specifications)

.BR

Bold alternating with Roman (especially useful for referring to other manual pages)

.I Italics

.IB Italics alternating with bold

.IR Italics alternating with Roman

.RB

Roman alternating with bold

- .RI Roman alternating with italics
- .SB Small alternating with bold
- .SM
 - Small (useful for acronyms)

Traditionally, each command can have up to six arguments, but the GNU implementation removes this limitation (you might still want to limit yourself to 6 arguments for portability's sake). Arguments are delimited by spaces. Double quotes can be used to specify an argument which contains spaces. For the macros that produce alternating type faces, the arguments will be printed next to each other without intervening spaces, so that the .BR command can be used to specify a word in bold followed by a mark of punctuation in Roman. If no arguments are given, the command is applied to the following line of text.

Other macros and strings

Below are other relevant macros and predefined strings. Unless noted otherwise, all macros cause a break (end the current line of text). Many of these macros set or use the "prevailing indent". The "prevailing indent" value is set by any macro with the parameter *i* below; macros may omit *i* in which case the current prevailing indent will be used. As a result, successive indented paragraphs can use the same indent without respecifying the indent value. A normal (nonindented) paragraph resets the prevailing indent value to its default value (0.5 inches). By default, a given indent is measured in ens; try to use ens or ems as units for indents, since these will automatically adjust to font size changes. The other key macro definitions are:

Normal paragraphs

- .LP Same as .PP (begin a new paragraph).
- .P Same as .PP (begin a new paragraph).
- .PP Begin a new paragraph and reset prevailing indent.

Relative margin indent

- .RS *i* Start relative margin indent: moves the left margin *i* to the right (if *i* is omitted, the prevailing indent value is used). A new prevailing indent is set to 0.5 inches. As a result, all following paragraph(s) will be indented until the corresponding .RE.
- .RE End relative margin indent and restores the previous value of the prevailing indent.

Indented paragraph macros

- .HP *i* Begin paragraph with a hanging indent (the first line of the paragraph is at the left margin of normal paragraphs, and the rest of the paragraph's lines are indented).
- .IP *x i* Indented paragraph with optional hanging tag. If the tag *x* is omitted, the entire following paragraph is indented by *i*. If the tag *x* is provided, it is hung at the left margin before the following indented paragraph (this is just like .TP except the tag is included with the command instead of being on the following line). If the tag is too long, the text after the tag will be moved down to the next line (text will not be lost or garbled). For bulleted lists, use this macro with \bu (bullet) or \em (em dash) as the tag, and for numbered lists, use the number or letter followed by a period as the tag; this simplifies translation to other formats.
- .TP *i* Begin paragraph with hanging tag. The tag is given on the next line, but its results are like those of the .IP command.

Hypertext link macros

- .UR *url* Insert a hypertext link to the URI (URL) *url*, with all text up to the following .UE macro as the link text.
- .UE [*trailer*] Terminate the link text of the preceding .UR macro, with the optional *trailer* (if present, usually a closing parenthesis and/or end-of-sentence punctuation) immediately following. For non-HTML output devices (e.g., man -Tutf8), the link text is followed by the URL in angle brackets; if there is no link text, the URL is printed as its own link text, surrounded by angle brackets. (Angle

brackets may not be available on all output devices.) For the HTML output device, the link text is hyperlinked to the URL; if there is no link text, the URL is printed as its own link text.

These macros have been supported since GNU Troff 1.20 (2009-01-05) and Heirloom DocTools Troff since 160217 (2016-02-17).

Miscellaneous macros

.DT	Reset tabs to default tab values (every 0.5 inches); does not cause a break.
.PD <i>d</i>	Set inter-paragraph vertical distance to <i>d</i> (if omitted, <i>d</i> =0.4v); does not cause a break.
.SS <i>t</i>	Subheading <i>t</i> (like .SH , but used for a subsection inside a section).

Predefined strings

The **man** package has the following predefined strings:

*R	Registration Symbol: ®
*S	Change to default font size
*(Tm	Trademark Symbol: ™
*(lq	Left angled double quote: “
*(rq	Right angled double quote: ”

Safe subset

Although technically **man** is a troff macro package, in reality a large number of other tools process man page files that don't implement all of troff's abilities. Thus, it's best to avoid some of troff's more exotic abilities where possible to permit these other tools to work correctly. Avoid using the various troff preprocessors (if you must, go ahead and use **tbl**(1), but try to use the **IP** and **TP** commands instead for two-column tables). Avoid using computations; most other tools can't process them. Use simple commands that are easy to translate to other formats. The following troff macros are believed to be safe (though in many cases they will be ignored by translators): **\", ., ad, bp, br, ce, de, ds, el, ie, if, fi, ft, hy, ig, in, na, ne, nf, nh, ps, so, sp, ti, tr**.

You may also use many troff escape sequences (those sequences beginning with \). When you need to include the backslash character as normal text, use \e. Other sequences you may use, where x or xx are any characters and N is any digit, include: \', \`, \-, \., \", \%, *x, *(xx, \xx, \\$N, \nx, \n(xx, \fx, and \f(xx. Avoid using the escape sequences for drawing graphics.

Do not use the optional parameter for **bp** (break page). Use only positive values for **sp** (vertical space). Don't define a macro (**de**) with the same name as a macro in this or the mdoc macro package with a different meaning; it's likely that such redefinitions will be ignored. Every positive indent (**in**) should be paired with a matching negative indent (although you should be using the **RS** and **RE** macros instead). The condition test (**if,ie**) should only have 't' or 'n' as the condition. Only translations (**tr**) that can be ignored should be used. Font changes (**ft** and the **\f** escape sequence) should only have the values 1, 2, 3, 4, R, I, B, P, or CW (the **ft** command may also have no parameters).

If you use capabilities beyond these, check the results carefully on several tools. Once you've confirmed that the additional capability is safe, let the maintainer of this document know about the safe command or sequence that should be added to this list.

FILES

/usr/share/groff/*/]tmac/an.tmac
/usr/man/whatis

NOTES

By all means include full URLs (or URIs) in the text itself; some tools such as **man2html**(1) can automatically turn them into hypertext links. You can also use the **UR** and **UE** macros to identify links to related information. If you include URLs, use the full URL (e.g.,<http://www.kernel.org>) to ensure that tools can automatically find the URLs.

Tools processing these files should open the file and examine the first nonwhitespace character. A period (.) or single quote (') at the beginning of a line indicates a troff-based file (such as man or mdoc). A left angle

bracket (<) indicates an SGML/XML-based file (such as HTML or Docbook). Anything else suggests simple ASCII text (e.g., a "catman" result).

Many man pages begin with '\>' followed by a space and a list of characters, indicating how the page is to be preprocessed. For portability's sake to non-troff translators we recommend that you avoid using anything other than **tbl**(1), and Linux can detect that automatically. However, you might want to include this information so your man page can be handled by other (less capable) systems. Here are the definitions of the preprocessors invoked by these characters:

- e** eqn(1)
- g** grap(1)
- p** pic(1)
- r** refer(1)
- t** tbl(1)
- v** vgrind(1)

BUGS

Most of the macros describe formatting (e.g., font type and spacing) instead of marking semantic content (e.g., this text is a reference to another page), compared to formats like mdoc and DocBook (even HTML has more semantic markings). This situation makes it harder to vary the **man** format for different media, to make the formatting consistent for a given media, and to automatically insert cross-references. By sticking to the safe subset described above, it should be easier to automate transitioning to a different reference page format in the future.

The Sun macro **TX** is not implemented.

SEE ALSO

apropos(1), **groff**(1), **lexgrog**(1), **man**(1), **man2html**(1), **whatis**(1), **groff_man**(7), **groff_www**(7),
man-pages(7), **mdoc**(7)

NAME

mandb – create or update the manual page index caches

SYNOPSIS

```
mandb [ -dqsupt?V ] [-C file] [manpath]
mandb [ -dqsut ] [-C file] -f filename ...
```

DESCRIPTION

mandb is used to initialise or manually update **index** database caches. The caches contain information relevant to the current state of the manual page system and the information stored within them is used by the man-db utilities to enhance their speed and functionality.

When creating or updating an **index**, **mandb** will warn of bad ROFF .so requests, bogus manual page file-names and manual pages from which the **whatis** cannot be parsed.

Supplying **mandb** with an optional colon-delimited path will override the internal system manual page hierarchy search path, determined from information found within the man-db configuration file.

DATABASE CACHES

mandb can be compiled with support for any one of the following database types.

Name	Async	Filename
Berkeley db	Yes	<i>index.bt</i>
GNU gdbm	Yes	<i>index.db</i>
UNIX ndbm	No	<i>index.(dir/pag)</i>

Those database types that support asynchronous updates provide enhanced speed at the cost of possible corruption in the event of unusual termination. In an unusual case where this has occurred, it may be necessary to rerun **mandb** with the **-c** option to re-create the databases from scratch.

OPTIONS**-d, --debug**

Print debugging information.

-q, --quiet

Produce no warnings.

-s, --no-straycats

Do not spend time looking for or adding information to the databases regarding stray cats.

-p, --no-purge

Do not spend time checking for deleted manual pages and purging them from the databases.

-c, --create

By default, **mandb** will try to update any previously created databases. If a database does not exist, it will create it. This option forces **mandb** to delete previous databases and re-create them from scratch, and implies **--no-purge**. This may be necessary if a database becomes corrupt or if a new database storage scheme is introduced in the future.

-u, --user-db

Create user databases only, even with write permissions necessary to create system databases.

-t, --test

Perform correctness checks on manual pages in the hierarchy search path. With this option, **mandb** will not alter existing databases.

-f, --filename

Update only the entries for the given filename. This option is not for general use; it is used internally by **man** when it has been compiled with the **MAN_DB_UPDATES** option and finds that a page is out of date. It implies **-p** and disables **-c** and **-s**.

-C file, --config-file=file

Use this user configuration file rather than the default of `~/.manpath`.

-?, --help

Show the usage message, then exit.

--usage

Print a short usage message and exit.

-V, --version

Show the version, then exit.

EXIT STATUS

0 Successful program execution.

1 Usage, syntax, or configuration file error.

2 Operational error.

3 A child process failed.

DIAGNOSTICS

The following warning messages can be emitted during database building.

<filename>: whatis parse for page(sec) failed

An attempt to extract whatis line(s) from the given `<filename>` failed. This is usually due to a poorly written manual page, but if many such messages are emitted it is likely that the system contains non-standard manual pages which are incompatible with the man-db whatis parser. See the **WHATIS PARSING** section in **lexgrog(1)** for more information.

<filename>: is a dangling symlink

`<filename>` does not exist but is referenced by a symbolic link. Further diagnostics are usually emitted to identify the `<filename>` of the offending link.

<filename>: bad symlink or ROFF ‘.so’ request

`<filename>` is either a symbolic link to, or contains a ROFF include request to, a non-existent file.

<filename>: ignoring bogus filename

The `<filename>` may or may not be a valid manual page but its name is invalid. This is usually due to a manual page with sectional extension `<x>` being put in manual page section `<y>`.

<filename_mask>: competing extensions

The wildcard `<filename_mask>` is not unique. This is usually caused by the existence of both a compressed and uncompressed version of the same manual page. All but the most recent are ignored.

FILES

`/etc/manpath.config`

man-db configuration file.

`/var/cache/man/index.(bt/db/dir/pag)`

An FHS compliant global *index* database cache.

Older locations for the database cache included:

`/usr/man/index.(bt/db/dir/pag)`

A traditional global *index* database cache.

`/var/catman/index.(bt/db/dir/pag)`

An alternate or FSSTND compliant global *index* database cache.

SEE ALSO

lexgrog(1), man(1), manpath(5), catman(8)

The **WHATIS PARSING** section formerly in this manual page is now part of **lexgrog(1)**.

AUTHOR

Wilf. (G.Wilford@ee.surrey.ac.uk).
Fabrizio Polacco (fpolacco@debian.org).
Colin Watson (cjwatson@debian.org).

BUGS

<https://gitlab.com/cjwatson/man-db/-/issues>
<https://savannah.nongnu.org/bugs/?group=man-db>

md(4) - Linux man page

Name

md - Multiple Device driver aka Linux Software RAID

Synopsis

`/dev/mdn`
`/dev/md/n`
`/dev/md/name`

Description

The **md** driver provides virtual devices that are created from one or more independent underlying devices. This array of devices often contains redundancy and the devices are often disk drives, hence the acronym RAID which stands for a Redundant Array of Independent Disks.

md supports RAID levels 1 (mirroring), 4 (striped array with parity device), 5 (striped array with distributed parity information), 6 (striped array with distributed dual redundancy information), and 10 (striped and mirrored). If some number of underlying devices fails while using one of these levels, the array will continue to function; this number is one for RAID levels 4 and 5, two for RAID level 6, and all but one ($N-1$) for RAID level 1, and dependent on configuration for level 10.

md also supports a number of pseudo RAID (non-redundant) configurations including RAID0 (striped array), LINEAR (catenated array), MULTIPATH (a set of different interfaces to the same device), and FAULTY (a layer over a single device into which errors can be injected).

Md Metadata

Each device in an array may have some *metadata* stored in the device. This metadata is sometimes called a **superblock**. The metadata records information about the structure and state of the array. This allows the array to be reliably re-assembled after a shutdown.

From Linux kernel version 2.6.10, **md** provides support for two different formats of metadata, and other formats can be added. Prior to this release, only one format is supported.

The common format -- known as version 0.90 -- has a superblock that is 4K long and is written into a 64K aligned block that starts at least 64K and less than 128K from the end of the device (i.e. to get the address of the superblock round the size of the device down to a multiple of 64K and then subtract 64K). The available size of each device is the amount of

space before the super block, so between 64K and 128K is lost when a device is incorporated into an MD array. This superblock stores multi-byte fields in a processor-dependent manner, so arrays cannot easily be moved between computers with different processors.

The new format -- known as version 1 -- has a superblock that is normally 1K long, but can be longer. It is normally stored between 8K and 12K from the end of the device, on a 4K boundary, though variations can be stored at the start of the device (version 1.1) or 4K from the start of the device (version 1.2). This metadata format stores multibyte data in a processor-independent format and supports up to hundreds of component devices (version 0.90 only supports 28).

The metadata contains, among other things:

LEVEL

The manner in which the devices are arranged into the array (linear, raid0, raid1, raid4, raid5, raid10, multipath).

UUID

a 128 bit Universally Unique Identifier that identifies the array that contains this device.

When a version 0.90 array is being reshaped (e.g. adding extra devices to a RAID5), the version number is temporarily set to 0.91. This ensures that if the reshape process is stopped in the middle (e.g. by a system crash) and the machine boots into an older kernel that does not support reshaping, then the array will not be assembled (which would cause data corruption) but will be left untouched until a kernel that can complete the reshape processes is used.

Arrays Without Metadata

While it is usually best to create arrays with superblocks so that they can be assembled reliably, there are some circumstances when an array without superblocks is preferred. These include:

LEGACY ARRAYS

Early versions of the **md** driver only supported Linear and Raid0 configurations and did not use a superblock (which is less critical with these configurations). While such arrays should be rebuilt with superblocks if possible, **md** continues to support them.

FAULTY

Being a largely transparent layer over a different device, the FAULTY personality doesn't gain anything from having a superblock.

MULTIPATH

It is often possible to detect devices which are different paths to the same storage directly rather than having a distinctive superblock written to the device and searched for on all paths. In this case, a MULTIPATH array with no superblock makes sense.

RAID1

In some configurations it might be desired to create a raid1 configuration that does not use a superblock, and to maintain the state of the array elsewhere. While not encouraged for general use, it does have special-purpose uses and is supported.

Arrays with External Metadata

From release 2.6.28, the *md* driver supports arrays with externally managed metadata. That is, the metadata is not managed by the kernel but rather by a user-space program which is external to the kernel. This allows support for a variety of metadata formats without cluttering the kernel with lots of details.

md is able to communicate with the user-space program through various sysfs attributes so that it can make appropriate changes to the metadata - for example to mark a device as faulty. When necessary, *md* will wait for the program to acknowledge the event by writing to a sysfs attribute. The manual page for [*mdmon*\(8\)](#) contains more detail about this interaction.

Containers

Many metadata formats use a single block of metadata to describe a number of different arrays which all use the same set of devices. In this case it is helpful for the kernel to know about the full set of devices as a whole. This set is known to *md* as a *container*. A container is an *md* array with externally managed metadata and with device offset and size so that it just covers the metadata part of the devices. The remainder of each device is available to be incorporated into various arrays.

Linear

A linear array simply catenates the available space on each drive to form one large virtual drive.

One advantage of this arrangement over the more common RAID0 arrangement is that the array may be reconfigured at a later time with an extra drive, so the array is made bigger without disturbing the data that is on the array. This can even be done on a live array.

If a chunksize is given with a LINEAR array, the usable space on each device is rounded down to a multiple of this chunksize.

Raid0

A RAID0 array (which has zero redundancy) is also known as a striped array. A RAID0 array is configured at creation with a **Chunk Size** which must be a power of two (prior to Linux 2.6.31), and at least 4 kibibytes.

The RAID0 driver assigns the first chunk of the array to the first device, the second chunk to the second device, and so on until all drives have been assigned one chunk. This collection of chunks forms a **stripe**. Further chunks are gathered into stripes in the same way, and are assigned to the remaining space in the drives.

If devices in the array are not all the same size, then once the smallest device has been exhausted, the RAID0 driver starts collecting chunks into smaller stripes that only span the drives which still have remaining space.

Raid1

A RAID1 array is also known as a mirrored set (though mirrors tend to provide reflected images, which RAID1 does not) or a plex.

Once initialised, each device in a RAID1 array contains exactly the same data. Changes are written to all devices in parallel. Data is read from any one device. The driver attempts to distribute read requests across all devices to maximise performance.

All devices in a RAID1 array should be the same size. If they are not, then only the amount of space available on the smallest device is used (any extra space on other devices is wasted).

Note that the read balancing done by the driver does not make the RAID1 performance profile be the same as for RAID0; a single stream of sequential input will not be accelerated (e.g. a single dd), but multiple sequential streams or a random workload will use more than one spindle. In theory, having an N-disk RAID1 will allow N sequential threads to read from all disks.

Individual devices in a RAID1 can be marked as "write-mostly". These drives are excluded from the normal read balancing and will only be read from when there is no other option. This can be useful for devices connected over a slow link.

Raid4

A RAID4 array is like a RAID0 array with an extra device for storing parity. This device is the last of the active devices in the array. Unlike RAID0, RAID4 also requires that all stripes span all drives, so extra space on devices that are larger than the smallest is wasted.

When any block in a RAID4 array is modified, the parity block for that stripe (i.e. the block in the parity device at the same device offset as the stripe) is also modified so that the parity block always contains the "parity" for the whole stripe. I.e. its content is equivalent to the result of performing an exclusive-or operation between all the data blocks in the stripe.

This allows the array to continue to function if one device fails. The data that was on that device can be calculated as needed from the parity block and the other data blocks.

Raid5

RAID5 is very similar to RAID4. The difference is that the parity blocks for each stripe, instead of being on a single device, are distributed across all devices. This allows more parallelism when writing, as two different block updates will quite possibly affect parity blocks on different devices so there is less contention.

This also allows more parallelism when reading, as read requests are distributed over all the devices in the array instead of all but one.

Raid6

RAID6 is similar to RAID5, but can handle the loss of any *two* devices without data loss. Accordingly, it requires N+2 drives to store N drives worth of data.

The performance for RAID6 is slightly lower but comparable to RAID5 in normal mode and single disk failure mode. It is very slow in dual disk failure mode, however.

Raid10

RAID10 provides a combination of RAID1 and RAID0, and is sometimes known as RAID1+0. Every datablock is duplicated some number of times, and the resulting collection of datablocks are distributed over multiple drives.

When configuring a RAID10 array, it is necessary to specify the number of replicas of each data block that are required (this will normally be 2) and whether the replicas should be 'near', 'offset' or 'far'. (Note that the 'offset' layout is only available from 2.6.18).

When 'near' replicas are chosen, the multiple copies of a given chunk are laid out consecutively across the stripes of the array, so the two copies of a datablock will likely be at the same offset on two adjacent devices.

When 'far' replicas are chosen, the multiple copies of a given chunk are laid out quite distant from each other. The first copy of all data blocks will be striped across the early part of all drives in RAID0 fashion, and then the next copy of all blocks will be striped across a later section of all drives, always ensuring that all copies of any given block are on different drives.

The 'far' arrangement can give sequential read performance equal to that of a RAID0 array, but at the cost of reduced write performance.

When 'offset' replicas are chosen, the multiple copies of a given chunk are laid out on consecutive drives and at consecutive offsets. Effectively each stripe is duplicated and the copies are offset by one device. This should give similar read characteristics to 'far' if a suitably large chunk size is used, but without as much seeking for writes.

It should be noted that the number of devices in a RAID10 array need not be a multiple of the number of replica of each data block; however, there must be at least as many devices as replicas.

If, for example, an array is created with 5 devices and 2 replicas, then space equivalent to 2.5 of the devices will be available, and every block will be stored on two different devices.

Finally, it is possible to have an array with both 'near' and 'far' copies. If an array is configured with 2 near copies and 2 far copies, then there will be a total of 4 copies of each block, each on a different drive. This is an artifact of the implementation and is unlikely to be of real value.

Multipath

MULTIPATH is not really a RAID at all as there is only one real device in a MULTIPATH md array. However there are multiple access points (paths) to this device, and one of these paths might fail, so there are some similarities.

A MULTIPATH array is composed of a number of logically different devices, often fibre channel interfaces, that all refer to the same real device. If one of these interfaces fails (e.g. due to cable problems), the multipath driver will attempt to redirect requests to another interface.

The MULTIPATH drive is not receiving any ongoing development and should be considered a legacy driver. The device-mapper based multipath drivers should be preferred for new installations.

Faulty

The FAULTY md module is provided for testing purposes. A faulty array has exactly one component device and is normally assembled without a superblock, so the md array created provides direct access to all of the data in the component device.

The FAULTY module may be requested to simulate faults to allow testing of other md levels or of filesystems. Faults can be chosen to trigger on read requests or write requests, and can be transient (a subsequent read/write at the address will probably succeed) or persistent (subsequent read/write of the same address will fail). Further, read faults can be "fixable" meaning that they persist until a write request at the same address.

Fault types can be requested with a period. In this case, the fault will recur repeatedly after the given number of requests of the relevant type. For example if persistent read faults have a period of 100, then every 100th read request would generate a fault, and the faulty sector would be recorded so that subsequent reads on that sector would also fail.

There is a limit to the number of faulty sectors that are remembered. Faults generated after this limit is exhausted are treated as transient.

The list of faulty sectors can be flushed, and the active list of failure modes can be cleared.

Unclean Shutdown

When changes are made to a RAID1, RAID4, RAID5, RAID6, or RAID10 array there is a possibility of inconsistency for short periods of time as each update requires at least two block to be written to different devices, and these writes probably won't happen at exactly the same time. Thus if a system with one of these arrays is shutdown in the middle of a write operation (e.g. due to power failure), the array may not be consistent.

To handle this situation, the md driver marks an array as "dirty" before writing any data to it, and marks it as "clean" when the array is being disabled, e.g. at shutdown. If the md driver finds an array to be dirty at startup, it proceeds to correct any possibly inconsistency. For RAID1, this involves copying the contents of the first drive onto all other drives. For RAID4, RAID5 and RAID6 this involves recalculating the parity for each stripe and making sure that the parity block has the correct data. For RAID10 it involves copying one of the replicas of each block onto all the others. This process, known as "resynchronising" or "resync" is performed in the background. The array can still be used, though possibly with reduced performance.

If a RAID4, RAID5 or RAID6 array is degraded (missing at least one drive, two for RAID6) when it is restarted after an unclean shutdown, it cannot recalculate parity, and so it is possible that data might be undetectably corrupted. The 2.4 md driver **does not** alert the operator to this condition. The 2.6 md driver will fail to start an array in this condition without manual intervention, though this behaviour can be overridden by a kernel parameter.

Recovery

If the md driver detects a write error on a device in a RAID1, RAID4, RAID5, RAID6, or RAID10 array, it immediately disables that device (marking it as faulty) and continues operation on the remaining devices. If there are spare drives, the driver will start recreating on one of the spare drives the data which was on that failed drive, either by copying a working drive in a RAID1 configuration, or by doing calculations with the parity block on RAID4, RAID5 or RAID6, or by finding and copying originals for RAID10.

In kernels prior to about 2.6.15, a read error would cause the same effect as a write error. In later kernels, a read-error will instead cause md to attempt a recovery by overwriting the bad block. i.e. it will find the correct data from elsewhere, write it over the block that failed, and then try to read it back again. If either the write or the re-read fail, md will treat the error the same way that a write error is treated, and will fail the whole device.

While this recovery process is happening, the md driver will monitor accesses to the array and will slow down the rate of recovery if other activity is happening, so that normal access to the array will not be unduly affected. When no other activity is happening, the recovery

process proceeds at full speed. The actual speed targets for the two different situations can be controlled by the **speed_limit_min** and **speed_limit_max** control files mentioned below.

Scrubbing and Mismatches

As storage devices can develop bad blocks at any time it is valuable to regularly read all blocks on all devices in an array so as to catch such bad blocks early. This process is called *scrubbing*.

md arrays can be scrubbed by writing either *check* or *repair* to the file *md/sync_action* in the *sysfs* directory for the device.

Requesting a scrub will cause *md* to read every block on every device in the array, and check that the data is consistent. For RAID1 and RAID10, this means checking that the copies are identical. For RAID4, RAID5, RAID6 this means checking that the parity block is (or blocks are) correct.

If a read error is detected during this process, the normal read-error handling causes correct data to be found from other devices and to be written back to the faulty device. In many case this will effectively *fix* the bad block.

If all blocks read successfully but are found to not be consistent, then this is regarded as a *mismatch*.

If *check* was used, then no action is taken to handle the mismatch, it is simply recorded. If *repair* was used, then a mismatch will be repaired in the same way that *resync* repairs arrays. For RAID5/RAID6 new parity blocks are written. For RAID1/RAID10, all but one block are overwritten with the content of that one block.

A count of mismatches is recorded in the *sysfs* file *md/mismatch_cnt*. This is set to zero when a scrub starts and is incremented whenever a sector is found that is a mismatch. *md* normally works in units much larger than a single sector and when it finds a mismatch, it does not determine exactly how many actual sectors were affected but simply adds the number of sectors in the IO unit that was used. So a value of 128 could simply mean that a single 64KB check found an error ($128 \times 512\text{bytes} = 64\text{KB}$).

If an array is created by *mdadm* with *--assume-clean* then a subsequent check could be expected to find some mismatches.

On a truly clean RAID5 or RAID6 array, any mismatches should indicate a hardware problem at some level - software issues should never cause such a mismatch.

However on RAID1 and RAID10 it is possible for software issues to cause a mismatch to be reported. This does not necessarily mean that the data on the array is corrupted. It could

simply be that the system does not care what is stored on that part of the array - it is unused space.

The most likely cause for an unexpected mismatch on RAID1 or RAID10 occurs if a swap partition or swap file is stored on the array.

When the swap subsystem wants to write a page of memory out, it flags the page as 'clean' in the memory manager and requests the swap device to write it out. It is quite possible that the memory will be changed while the write-out is happening. In that case the 'clean' flag will be found to be clear when the write completes and so the swap subsystem will simply forget that the swapout had been attempted, and will possibly choose a different page to write out.

If the swap device was on RAID1 (or RAID10), then the data is sent from memory to a device twice (or more depending on the number of devices in the array). Thus it is possible that the memory gets changed between the times it is sent, so different data can be written to the different devices in the array. This will be detected by *check* as a mismatch. However it does not reflect any corruption as the block where this mismatch occurs is being treated by the swap system as being empty, and the data will never be read from that block.

It is conceivable for a similar situation to occur on non-swap files, though it is less likely.

Thus the *mismatch_cnt* value can not be interpreted very reliably on RAID1 or RAID10, especially when the device is used for swap.

Bitmap Write-intent Logging

From Linux 2.6.13, *md* supports a bitmap based write-intent log. If configured, the bitmap is used to record which blocks of the array may be out of sync. Before any write request is honoured, *md* will make sure that the corresponding bit in the log is set. After a period of time with no writes to an area of the array, the corresponding bit will be cleared.

This bitmap is used for two optimisations.

Firstly, after an unclean shutdown, the resync process will consult the bitmap and only resync those blocks that correspond to bits in the bitmap that are set. This can dramatically reduce resync time.

Secondly, when a drive fails and is removed from the array, *md* stops clearing bits in the intent log. If that same drive is re-added to the array, *md* will notice and will only recover the sections of the drive that are covered by bits in the intent log that are set. This can allow a device to be temporarily removed and reinserted without causing an enormous recovery cost.

The intent log can be stored in a file on a separate device, or it can be stored near the superblocks of an array which has superblocks.

It is possible to add an intent log to an active array, or remove an intent log if one is present.

In 2.6.13, intent bitmaps are only supported with RAID1. Other levels with redundancy are supported from 2.6.15.

Write-behind

From Linux 2.6.14, *md* supports WRITE-BEHIND on RAID1 arrays.

This allows certain devices in the array to be flagged as *write-mostly*. MD will only read from such devices if there is no other option.

If a write-intent bitmap is also provided, write requests to write-mostly devices will be treated as write-behind requests and md will not wait for writes to those requests to complete before reporting the write as complete to the filesystem.

This allows for a RAID1 with WRITE-BEHIND to be used to mirror data over a slow link to a remote computer (providing the link isn't too slow). The extra latency of the remote link will not slow down normal operations, but the remote system will still have a reasonably up-to-date copy of all data.

Restriping

Restriping, also known as *Reshaping*, is the processes of re-arranging the data stored in each stripe into a new layout. This might involve changing the number of devices in the array (so the stripes are wider), changing the chunk size (so stripes are deeper or shallower), or changing the arrangement of data and parity (possibly changing the raid level, e.g. 1 to 5 or 5 to 6).

As of Linux 2.6.35, *md* can reshape a RAID4, RAID5, or RAID6 array to have a different number of devices (more or fewer) and to have a different layout or chunk size. It can also convert between these different RAID levels. It can also convert between RAID0 and RAID10, and between RAID0 and RAID4 or RAID5. Other possibilities may follow in future kernels.

During any stripe process there is a 'critical section' during which live data is being overwritten on disk. For the operation of increasing the number of drives in a raid5, this critical section covers the first few stripes (the number being the product of the old and new number of devices). After this critical section is passed, data is only written to areas of the array which no longer hold live data -- the live data has already been located away.

For a reshape which reduces the number of devices, the 'critical section' is at the end of the reshape process.

md is not able to ensure data preservation if there is a crash (e.g. power failure) during the critical section. If md is asked to start an array which failed during a critical section of restriping, it will fail to start the array.

To deal with this possibility, a user-space program must

- Disable writes to that section of the array (using the **sysfs** interface),
- take a copy of the data somewhere (i.e. make a backup),
- allow the process to continue and invalidate the backup and restore write access once the critical section is passed, and
- provide for restoring the critical data before restarting the array after a system crash.

mdadm versions from 2.4 do this for growing a RAID5 array.

For operations that do not change the size of the array, like simply increasing chunk size, or converting RAID5 to RAID6 with one extra device, the entire process is the critical section. In this case, the stripe will need to progress in stages, as a section is suspended, backed up, restriped, and released.

Sysfs Interface

Each block device appears as a directory in *sysfs* (which is usually mounted at **/sys**). For MD devices, this directory will contain a subdirectory called **md** which contains various files for providing access to information about the array.

This interface is documented more fully in the file **Documentation/md.txt** which is distributed with the kernel sources. That file should be consulted for full documentation. The following are just a selection of attribute files that are available.

md/sync_speed_min

This value, if set, overrides the system-wide setting in **/proc/sys/dev/raid/speed_limit_min** for this array only. Writing the value **system** to this file will cause the system-wide setting to have effect.

md/sync_speed_max

This is the partner of **md/sync_speed_min** and overrides **/proc/sys/dev/raid/speed_limit_max** described below.

md/sync_action

This can be used to monitor and control the resync/recovery process of MD. In particular, writing "check" here will cause the array to read all data block and check that they are consistent (e.g. parity is correct, or all mirror replicas are the same). Any discrepancies found are **NOT** corrected.

A count of problems found will be stored in **md/mismatch_count**.

Alternately, "repair" can be written which will cause the same check to be performed, but any errors will be corrected.

Finally, "idle" can be written to stop the check/repair process.

md/stripe_cache_size

This is only available on RAID5 and RAID6. It records the size (in pages per device) of the stripe cache which is used for synchronising all write operations to the array and all read operations if the array is degraded. The default is 256. Valid values are 17 to 32768. Increasing this number can increase performance in some situations, at some cost in system memory. Note, setting this value too high can result in an "out of memory" condition for the system.

`memory_consumed = system_page_size * nr_disks * stripe_cache_size`

md/preread_bypass_threshold

This is only available on RAID5 and RAID6. This variable sets the number of times MD will service a full-stripe-write before servicing a stripe that requires some "prereading". For fairness this defaults to 1. Valid values are 0 to `stripe_cache_size`. Setting this to 0 maximizes sequential-write throughput at the cost of fairness to threads doing small or random writes.

Kernel Parameters

The md driver recognised several different kernel parameters.

raid=noautodetect

This will disable the normal detection of md arrays that happens at boot time. If a drive is partitioned with MS-DOS style partitions, then if any of the 4 main partitions has a partition type of 0xFD, then that partition will normally be inspected to see if it is part of an MD array, and if any full arrays are found, they are started. This kernel parameter disables this behaviour.

raid=partitionable

raid=part

These are available in 2.6 and later kernels only. They indicate that autodetected MD arrays should be created as partitionable arrays, with a different major device number to the original non-partitionable md arrays. The device number is listed as `mdp` in `/proc/devices`.

md_mod.start_ro=1

/sys/module/md_mod/parameters/start_ro

This tells md to start all arrays in read-only mode. This is a soft read-only that will automatically switch to read-write on the first write request. However until that write request, nothing is written to any device by md, and in particular, no resync or recovery operation is started.

md_mod.start_dirty_degraded=1

/sys/module/md_mod/parameters/start_dirty_degraded

As mentioned above, md will not normally start a RAID4, RAID5, or RAID6 that is both dirty and degraded as this situation can imply hidden data loss. This can be awkward if the root filesystem is affected. Using this module parameter allows such arrays to be started at boot time. It should be understood that there is a real (though small) risk of data corruption in this situation.

md=n,dev,dev,...

md=dn,dev,dev,...

This tells the md driver to assemble **/dev/md n** from the listed devices. It is only necessary to start the device holding the root filesystem this way. Other arrays are best started once the system is booted.

In 2.6 kernels, the **d** immediately after the **=** indicates that a partitionable device (e.g. **/dev/md/d0**) should be created rather than the original non-partitionable device.

md=n,l,c,i,dev...

This tells the md driver to assemble a legacy RAID0 or LINEAR array without a superblock. *n* gives the md device number, *l* gives the level, 0 for RAID0 or -1 for LINEAR, *c* gives the chunk size as a base-2 logarithm offset by twelve, so 0 means 4K, 1 means 8K. *i* is ignored (legacy support).

Files

/proc/mdstat

Contains information about the status of currently running array.

/proc/sys/dev/raid/speed_limit_min

A readable and writable file that reflects the current "goal" rebuild speed for times when non-rebuild activity is current on an array. The speed is in Kibibytes per second, and is a per-device rate, not a per-array rate (which means that an array with more disks will shuffle more data for a given speed). The default is 1000.

/proc/sys/dev/raid/speed_limit_max

A readable and writable file that reflects the current "goal" rebuild speed for times when no non-rebuild activity is current on an array. The default is 200,000.

See Also

[mdadm\(8\)](#), **[mkraid\(8\)](#)**.

Referenced By

[mdadm.conf\(5\)](#), **[mdassemble\(8\)](#)**, **[xfs_info\(8\)](#)**

NAME

md5sum – compute and check MD5 message digest

SYNOPSIS

md5sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check MD5 (128-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read MD5 sums from the FILEs and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in RFC 1321. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

BUGS

Do not use the MD5 algorithm for security related purposes. Instead, use an SHA-2 algorithm, implemented in the programs sha224sum(1), sha256sum(1), sha384sum(1), sha512sum(1), or the BLAKE2 algorithm, implemented in b2sum(1)

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/md5sum>>
or available locally via: info '(coreutils) md5sum invocation'

NAME

`mdadm` – manage MD devices *aka* Linux Software RAID

SYNOPSIS

`mdadm [mode] <raiddevice> [options] <component-devices>`

DESCRIPTION

RAID devices are virtual devices created from two or more real block devices. This allows multiple devices (typically disk drives or partitions thereof) to be combined into a single device to hold (for example) a single filesystem. Some RAID levels include redundancy and so can survive some degree of device failure.

Linux Software RAID devices are implemented through the md (Multiple Devices) device driver.

Currently, Linux supports **LINEAR** md devices, **RAID0** (striping), **RAID1** (mirroring), **RAID4**, **RAID5**, **RAID6**, **RAID10**, **MULTIPATH**, **FAULTY**, and **CONTAINER**.

MULTIPATH is not a Software RAID mechanism, but does involve multiple devices: each device is a path to one common physical storage device. New installations should not use md/multipath as it is not well supported and has no ongoing development. Use the Device Mapper based multipath-tools instead.

FAULTY is also not true RAID, and it only involves one device. It provides a layer over a true device that can be used to inject faults.

CONTAINER is different again. **ACONTAINER** is a collection of devices that are managed as a set. This is similar to the set of devices connected to a hardware RAID controller. The set of devices may contain a number of different RAID arrays each utilising some (or all) of the blocks from a number of the devices in the set. For example, two devices in a 5-device set might form a RAID1 using the whole devices. The remaining three might have a RAID5 over the first half of each device, and a RAID0 over the second half.

With a **CONTAINER**, there is one set of metadata that describes all of the arrays in the container. So when `mdadm` creates a **CONTAINER** device, the device just represents the metadata. Other normal arrays (RAID1 etc) can be created inside the container.

MODES

`mdadm` has several major modes of operation:

Assemble

Assemble the components of a previously created array into an active array. Components can be explicitly given or can be searched for. `mdadm` checks that the components do form a bona fide array, and can, on request, fiddle superblock information so as to assemble a faulty array.

Build Build an array that doesn't have per-device metadata (superblocks). For these sorts of arrays, `mdadm` cannot differentiate between initial creation and subsequent assembly of an array. It also cannot perform any checks that appropriate components have been requested. Because of this, the **Build** mode should only be used together with a complete understanding of what you are doing.

Create Create a new array with per-device metadata (superblocks). Appropriate metadata is written to each device, and then the array comprising those devices is activated. A 'resync' process is started to make sure that the array is consistent (e.g. both sides of a mirror contain the same data) but the content of the device is left otherwise untouched. The array can be used as soon as it has been created. There is no need to wait for the initial resync to finish.

Follow or Monitor

Monitor one or more md devices and act on any state changes. This is only meaningful for RAID1, 4, 5, 6, 10 or multipath arrays, as only these have interesting state. RAID0 or Linear never have missing, spare, or failed drives, so there is nothing to monitor.

- Grow** Grow (or shrink) an array, or otherwise reshape it in some way. Currently supported growth options including changing the active size of component devices and changing the number of active devices in Linear and RAID levels 0/1/4/5/6, changing the RAID level between 0, 1, 5, and 6, and between 0 and 10, changing the chunk size and layout for RAID 0,4,5,6,10 as well as adding or removing a write-intent bitmap and changing the array's consistency policy.

Incremental Assembly

Add a single device to an appropriate array. If the addition of the device makes the array runnable, the array will be started. This provides a convenient interface to a *hot-plug* system. As each device is detected, *mdadm* has a chance to include it in some array as appropriate. Optionally, when the *--fail* flag is passed in we will remove the device from any active array instead of adding it.

If a **CONTAINER** is passed to *mdadm* in this mode, then any arrays within that container will be assembled and started.

Manage

This is for doing things to specific components of an array such as adding new spares and removing faulty devices.

- Misc** This is an 'everything else' mode that supports operations on active arrays, operations on component devices such as erasing old superblocks, and information gathering operations.

Auto-detect

This mode does not act on a specific device or array, but rather it requests the Linux Kernel to activate any auto-detected arrays.

OPTIONS**Options for selecting a mode are:****-A, --assemble**

Assemble a pre-existing array.

-B, --build

Build a legacy array without superblocks.

-C, --create

Create a new array.

-F, --follow, --monitor

Select **Monitor** mode.

-G, --grow

Change the size or shape of an active array.

-I, --incremental

Add/remove a single device to/from an appropriate array, and possibly start the array.

--auto-detect

Request that the kernel starts any auto-detected arrays. This can only work if *md* is compiled into the kernel — not if it is a module. Arrays can be auto-detected by the kernel if all the components are in primary MS-DOS partitions with partition type **FD**, and all use v0.90 metadata. In-kernel autodetect is not recommended for new installations. Using *mdadm* to detect and assemble arrays — possibly in *aninitrd* — is substantially more flexible and should be preferred.

If a device is given before any options, or if the first option is one of **--add**, **--re-add**, **--add-spare**, **--fail**, **--remove**, or **--replace**, then the **MANAGE** mode is assumed. Anything other than these will cause the **Misc** mode to be assumed.

Options that are not mode-specific are:**-h, --help**

Display general help message or, after one of the above options, a mode-specific help message.

--help-options

Display more detailed help about command line parsing and some commonly used options.

-V, --version

Print version information for *mdadm*.

-v, --verbose

Be more verbose about what is happening. This can be used twice to be extra-verbose. The extra verbosity currently only affects **--detail** **--scan** and **--examine** **--scan**.

-q, --quiet

Avoid printing purely informative messages. With this, *mdadm* will be silent unless there is something really important to report.

-f, --force

Be more forceful about certain operations. See the various modes for the exact meaning of this option in different contexts.

-c, --config=

Specify the config file or directory. Default is to use **/etc/mdadm/mdadm.conf** and **/etc/mdadm/mdadm.conf.d**, or if those are missing then **/etc/mdadm.conf** and **/etc/mdadm.conf.d**. If the config file given is **partitions** then nothing will be read, but *mdadm* will act as though the config file contained exactly

DEVICE partitions containers

and will read **/proc/partitions** to find a list of devices to scan, and **/proc/mdstat** to find a list of containers to examine. If the word **none** is given for the config file, then *mdadm* will act as though the config file were empty.

If the name given is of a directory, then *mdadm* will collect all the files contained in the directory with a name ending in **.conf**, sort them lexically, and process all of those files as config files.

-s, --scan

Scan config file or **/proc/mdstat** for missing information. In general, this option gives *mdadm* permission to get any missing information (like component devices, array devices, array identities, and alert destination) from the configuration file (see previous option); one exception is MISC

mode when using **--detail** or **--stop**, in which case **--scan** says to get a list of array devices from **/proc/mdstat**.

-e, --metadata=

Declare the style of RAID metadata (superblock) to be used. The default is 1.2 for **--create**, and to guess for other operations. The default can be overridden by setting the **metadata** value for the **CREATE** keyword in **mdadm.conf**.

Options are:

0, 0.90 Use the original 0.90 format superblock. This format limits arrays to 28 component devices and limits component devices of levels 1 and greater to 2 terabytes. It is also possible for there to be confusion about whether the superblock applies to a whole device or just the last partition, if that partition starts on a 64K boundary.

1, 1.0, 1.1, 1.2 default

Use the new version-1 format superblock. This has fewer restrictions. It can easily be moved between hosts with different endian-ness, and a recovery operation can be check-pointed and restarted. The different sub-versions store the superblock at different locations on the device, either at the end (for 1.0), at the start (for 1.1) or 4K from the start (for 1.2). "1" is equivalent to "1.2" (the commonly preferred 1.x format). "default" is equivalent to "1.2".

ddf Use the "Industry Standard" DDF (Disk Data Format) format defined by SNIA. When creating a DDF array a **CONTAINER** will be created, and normal arrays can be created in that container.

imsm Use the Intel(R) Matrix Storage Manager metadata format. This creates a **CONTAINER** which is managed in a similar manner to DDF, and is supported by an option-rom on some platforms:

<https://www.intel.com/content/www/us/en/support/products/122484/memory-and-storage/ssd-software/intel-virtual-raid-on-cpu-intel-vroc.html>

--homehost=

This will override any **HOMEHOST** setting in the config file and provides the identity of the host which should be considered the home for any arrays.

When creating an array, the **homehost** will be recorded in the metadata. For version-1 superblocks, it will be prefixed to the array name. For version-0.90 superblocks, part of the SHA1 hash of the hostname will be stored in the later half of the UUID.

When reporting information about an array, any array which is tagged for the given homehost will be reported as such.

When using Auto-Assemble, only arrays tagged for the given homehost will be allowed to use 'local' names (i.e. not ending in '_' followed by a digit string). See below under **Auto Assembly**.

The special name "**any**" can be used as a wild card. If an array is created with **--homehost=any** then the name "**any**" will be stored in the array and it can be assembled in the same way on any host. If an array is assembled with this option, then the homehost recorded on the array will be ignored.

--prefer=

When *mdadm* needs to print the name for a device it normally finds the name in */dev* which refers to the device and is shortest. When a path component is given with **--prefer** *mdadm* will prefer a longer name if it contains that component. For example **--prefer=by-uuid** will prefer a name in a subdirectory of */dev* called **by-uuid**.

This functionality is currently only provided by **--detail** and **--monitor**.

--home-cluster=

specifies the cluster name for the md device. The md device can be assembled only on the cluster which matches the name specified. If this option is not provided, *mdadm* tries to detect the cluster name automatically.

For create, build, or grow:**-n, --raid-devices=**

Specify the number of active devices in the array. This, plus the number of spare devices (see below) must equal the number of *component-devices* (including "missing" devices) that are listed on the command line for **--create**. Setting a value of 1 is probably a mistake and so requires that **--force** be specified first. A value of 1 will then be allowed for linear, multipath, RAID0 and RAID1. It is never allowed for RAID4, RAID5 or RAID6.

This number can only be changed using **--grow** for RAID1, RAID4, RAID5 and RAID6 arrays, and only on kernels which provide the necessary support.

-x, --spare-devices=

Specify the number of spare (eXtra) devices in the initial array. Spares can also be added and removed later. The number of component devices listed on the command line must equal the number of RAID devices plus the number of spare devices.

-z, --size=

Amount (in Kilobytes) of space to use from each drive in RAID levels 1/4/5/6. This must be a multiple of the chunk size, and must leave about 128Kb of space at the end of the drive for the RAID superblock. When specified as *max*, (as it often is) the smallest drive (or partition) sets the size. In that case, a warning will follow if the drives, as a group, have sizes that differ by more than one percent.

A suffix of 'K', 'M', 'G' or 'T' can be given to indicate Kilobytes, Megabytes, Gigabytes or Terabytes respectively.

Sometimes a replacement drive can be a little smaller than the original drives though this should be minimised by IDEMA standards. Such a replacement drive will be rejected by *md*. To guard against this it can be useful to set the initial size slightly smaller than the smaller device with the aim that it will still be larger than any replacement.

This value can be set with **--grow** for RAID level 1/4/5/6 though DDF arrays may not be able to support this. If the array was created with a size smaller than the currently active drives, the extra space can be accessed using **--grow**. The size can be given as **max** which means to choose the largest size that fits on all current drives.

Before reducing the size of the array (with **--grow --size=**) you should make sure that space isn't needed. If the device holds a filesystem, you would need to resize the filesystem to use less space.

After reducing the array size you should check that the data stored in the device is still available. If the device holds a filesystem, then an 'fsck' of the filesystem is a minimum requirement. If

there are problems the array can be made bigger again with no loss with another **--grow --size=** command.

This value cannot be used when creating a **CONTAINER** such as with DDF and IMSM metadata, though it perfectly valid when creating an array inside a container.

-Z, --array-size=

This is only meaningful with **--grow** and its effect is not persistent: when the array is stopped and restarted the default array size will be restored.

Setting the array-size causes the array to appear smaller to programs that access the data. This is particularly needed before reshaping an array so that it will be smaller. As the reshape is not reversible, but setting the size with **--array-size** is, it is required that the array size is reduced as appropriate before the number of devices in the array is reduced.

Before reducing the size of the array you should make sure that space isn't needed. If the device holds a filesystem, you would need to resize the filesystem to use less space.

After reducing the array size you should check that the data stored in the device is still available. If the device holds a filesystem, then an 'fsck' of the filesystem is a minimum requirement. If there are problems the array can be made bigger again with no loss with another **--grow --array-size=** command.

A suffix of 'K', 'M', 'G' or 'T' can be given to indicate Kilobytes, Megabytes, Gigabytes or Terabytes respectively. A value of **max** restores the apparent size of the array to be whatever the real amount of available space is.

Clustered arrays do not support this parameter yet.

-c, --chunk=

Specify chunk size of kilobytes. The default when creating an array is 512KB. To ensure compatibility with earlier versions, the default when building an array with no persistent metadata is 64KB. This is only meaningful for RAID0, RAID4, RAID5, RAID6, and RAID10.

RAID4, RAID5, RAID6, and RAID10 require the chunk size to be a power of 2. In any case it must be a multiple of 4KB.

A suffix of 'K', 'M', 'G' or 'T' can be given to indicate Kilobytes, Megabytes, Gigabytes or Terabytes respectively.

--rounding=

Specify rounding factor for a Linear array. The size of each component will be rounded down to a multiple of this size. This is a synonym for **--chunk** but highlights the different meaning for Linear as compared to other RAID levels. The default is 64K if a kernel earlier than 2.6.16 is in use, and is 0K (i.e. no rounding) in later kernels.

-l, --level=

Set RAID level. When used with **--create**, options are: linear, raid0, 0, stripe, raid1, 1, mirror, raid4, 4, raid5, 5, raid6, 6, raid10, 10, multipath, mp, faulty, container. Obviously some of these are synonymous.

When a **CONTAINER** metadata type is requested, only the **container** level is permitted, and it does not need to be explicitly given.

When used with **--build**, only linear, stripe, raid0, 0, raid1, multipath, mp, and faulty are valid.

Can be used with **--grow** to change the RAID level in some cases. See LEVEL CHANGES below.

-p, --layout=

This option configures the fine details of data layout for RAID5, RAID6, and RAID10 arrays, and controls the failure modes for *faulty*. It can also be used for working around a kernel bug with RAID0, but generally doesn't need to be used explicitly.

The layout of the RAID5 parity block can be one of **left-asymmetric**, **left-symmetric**, **right-asymmetric**, **right-symmetric**, **la**, **ra**, **ls**, **rs**. The default is **left-symmetric**.

It is also possible to cause RAID5 to use a RAID4-like layout by choosing **parity-first**, or **parity-last**.

Finally for RAID5 there are DDF-compatible layouts, **ddf-zero-restart**, **ddf-N-restart**, and **ddf-N-continue**.

These same layouts are available for RAID6. There are also 4 layouts that will provide an intermediate stage for converting between RAID5 and RAID6. These provide a layout which is identical to the corresponding RAID5 layout on the first N-1 devices, and has the 'Q' syndrome (the second 'parity' block used by RAID6) on the last device. These layouts are:**left-symmetric-6**, **right-symmetric-6**, **left-asymmetric-6**, **right-asymmetric-6**, and **parity-first-6**.

When setting the failure mode for level *faulty*, the options are: **write-transient**, **wt**, **read-transient**, **rt**, **write-persistent**, **wp**, **read-persistent**, **rp**, **write-all**, **read-fixable**, **rf**, **clear**, **flush**, **none**.

Each failure mode can be followed by a number, which is used as a period between fault generation. Without a number, the fault is generated once on the first relevant request. With a number, the fault will be generated after that many requests, and will continue to be generated every time the period elapses.

Multiple failure modes can be current simultaneously by using the **--grow** option to set subsequent failure modes.

"clear" or "none" will remove any pending or periodic failure modes, and "flush" will clear any persistent faults.

The layout options for RAID10 are one of 'n', 'o' or 'f' followed by a small number. The default is 'n2'. The supported options are:

'n' signals 'near' copies. Multiple copies of one data block are at similar offsets in different devices.

'o' signals 'offset' copies. Rather than the chunks being duplicated within a stripe, whole stripes are duplicated but are rotated by one device so duplicate blocks are on different devices. Thus subsequent copies of a block are in the next drive, and are one chunk further down.

'f' signals 'far' copies (multiple copies have very different offsets). See md(4) for more detail about 'near', 'offset', and 'far'.

The number is the number of copies of each datablock. 2 is normal, 3 can be useful. This number

can be at most equal to the number of devices in the array. It does not need to divide evenly into that number (e.g. it is perfectly legal to have an 'n2' layout for an array with an odd number of devices).

A bug introduced in Linux 3.14 means that RAID0 arrays **with devices of differing sizes** started using a different layout. This could lead to data corruption. Since Linux 5.4 (and various stable releases that received backports), the kernel will not accept such an array unless a layout is explicitly set. It can be set to '**original**' or '**alternate**'. When creating a new array, *mdadm* will select '**original**' by default, so the layout does not normally need to be set. An array created for either '**original**' or '**alternate**' will not be recognized by an (unpatched) kernel prior to 5.4. To create a RAID0 array with devices of differing sizes that can be used on an older kernel, you can set the layout to '**dangerous**'. This will use whichever layout the running kernel supports, so the data on the array may become corrupt when changing kernel from pre-3.14 to a later kernel.

When an array is converted between RAID5 and RAID6 an intermediate RAID6 layout is used in which the second parity block (Q) is always on the last device. To convert a RAID5 to RAID6 and leave it in this new layout (which does not require re-striping) use **--layout=preserve**. This will try to avoid any restriping.

The converse of this is **--layout=normalise** which will change a non-standard RAID6 layout into a more standard arrangement.

--parity=

same as **--layout** (thus explaining the p of **-p**).

-b, --bitmap=

Specify a file to store a write-intent bitmap in. The file should not exist unless **--force** is also given. The same file should be provided when assembling the array. If the word **internal** is given, then the bitmap is stored with the metadata on the array, and so is replicated on all devices. If the word **none** is given with **--grow** mode, then any bitmap that is present is removed. If the word **clustered** is given, the array is created for a clustered environment. One bitmap is created for each node as defined by the **--nodes** parameter and are stored internally.

To help catch typing errors, the filename must contain at least one slash ('/') if it is a real file (not 'internal' or 'none').

Note: external bitmaps are only known to work on ext2 and ext3. Storing bitmap files on other filesystems may result in serious problems.

When creating an array on devices which are 100G or larger, *mdadm* automatically adds an internal bitmap as it will usually be beneficial. This can be suppressed with **--bitmap=none** or by selecting a different consistency policy with **--consistency-policy**.

--bitmap-chunk=

Set the chunksize of the bitmap. Each bit corresponds to that many Kilobytes of storage. When using a file based bitmap, the default is to use the smallest size that is at-least 4 and requires no more than 2^{21} chunks. When using an **internal** bitmap, the chunksize defaults to 64Meg, or larger if necessary to fit the bitmap into the available space.

A suffix of 'K', 'M', 'G' or 'T' can be given to indicate Kilobytes, Megabytes, Gigabytes or Terabytes respectively.

-W, --write-mostly

subsequent devices listed in a **--build**, **--create**, or **--add** command will be flagged as 'write-mostly'. This is valid for RAID1 only and means that the 'md' driver will avoid reading from these devices if at all possible. This can be useful if mirroring over a slow link.

--write-behind=

Specify that write-behind mode should be enabled (valid for RAID1 only). If an argument is specified, it will set the maximum number of outstanding writes allowed. The default value is 256. A write-intent bitmap is required in order to use write-behind mode, and write-behind is only attempted on drives marked as *write-mostly*.

--failfast

subsequent devices listed in a **--create** or **--add** command will be flagged as 'failfast'. This is valid for RAID1 and RAID10 only. IO requests to these devices will be encouraged to fail quickly rather than cause long delays due to error handling. Also no attempt is made to repair a read error on these devices.

If an array becomes degraded so that the 'failfast' device is the only usable device, the 'failfast' flag will then be ignored and extended delays will be preferred to complete failure.

The 'failfast' flag is appropriate for storage arrays which have a low probability of true failure, but which may sometimes cause unacceptable delays due to internal maintenance functions.

--assume-clean

Tell *mdadm* that the array pre-existed and is known to be clean. It can be useful when trying to recover from a major failure as you can be sure that no data will be affected unless you actually write to the array. It can also be used when creating a RAID1 or RAID10 if you want to avoid the initial resync, however this practice — while normally safe — is not recommended. Use this only if you really know what you are doing.

When the devices that will be part of a new array were filled with zeros before creation the operator knows the array is actually clean. If that is the case, such as after running badblocks, this argument can be used to tell *mdadm* the facts the operator knows.

When an array is resized to a larger size with **--grow --size=** the new space is normally resynced in that same way that the whole array is resynced at creation. From Linux version 3.0, **--assume-clean** can be used with that command to avoid the automatic resync.

--backup-file=

This is needed when **--grow** is used to increase the number of raid-devices in a RAID5 or RAID6 if there are no spare devices available, or to shrink, change RAID level or layout. See the GROW MODE section below on RAID-DEVICES CHANGES. The file must be stored on a separate device, not on the RAID array being reshaped.

--data-offset=

Arrays with 1.x metadata can leave a gap between the start of the device and the start of array data. This gap can be used for various metadata. The start of data is known as the *data-offset*. Normally an appropriate data offset is computed automatically. However it can be useful to set it explicitly such as when re-creating an array which was originally created using a different version of *mdadm* which computed a different offset.

Setting the offset explicitly over-rides the default. The value given is in Kilobytes unless a suffix of 'K', 'M', 'G' or 'T' is used to explicitly indicate Kilobytes, Megabytes, Gigabytes or Terabytes respectively.

Since Linux 3.4, **--data-offset** can also be used with **--grow** for some RAID levels (initially on RAID10). This allows the data-offset to be changed as part of the reshape process. When the data offset is changed, no backup file is required as the difference in offsets is used to provide the same functionality.

When the new offset is earlier than the old offset, the number of devices in the array cannot shrink. When it is after the old offset, the number of devices in the array cannot increase.

When creating an array, **--data-offset** can be specified as **variable**. In the case each member device is expected to have a offset appended to the name, separated by a colon. This makes it possible to recreate exactly an array which has varying data offsets (as can happen when different versions of *mdadm* are used to add different devices).

--continue

This option is complementary to the **--freeze-reshape** option for assembly. It is needed when **--grow** operation is interrupted and it is not restarted automatically due to **--freeze-reshape** usage during array assembly. This option is used together with **-G** , (**--gr ow**) command and device for a pending reshape to be continued. All parameters required for reshape continuation will be read from array metadata. If initial **--grow** command had required **--backup-file=** option to be set, continuation option will require to have exactly the same backup file given as well.

Any other parameter passed together with **--continue** option will be ignored.

-N, --name=

Set a **name** for the array. This is currently only effective when creating an array with a version-1 superblock, or an array in a DDF container. The name is a simple textual string that can be used to identify array components when assembling. If name is needed but not specified, it is taken from the basename of the device that is being created. e.g. when creating */dev/md/home* the **name** will default to *home*. (Does not work in Grow mode.)

-R, --run

Insist that *mdadm* run the array, even if some of the components appear to be active in another array or filesystem. Normally *mdadm* will ask for confirmation before including such components in an array. This option causes that question to be suppressed.

-f, --force

Insist that *mdadm* accept the geometry and layout specified without question. Normally *mdadm* will not allow creation of an array with only one device, and will try to create a RAID5 array with one missing drive (as this makes the initial resync work faster). With **--force**, *mdadm* will not try to be so clever.

-o, --readonly

Start the array **read only** rather than read-write as normal. No writes will be allowed to the array, and no resync, recovery, or reshape will be started. It works with Create, Assemble, Manage and Misc mode.

-a, --auto{=yes,md,mdp,part,p}{NN}

Instruct *mdadm* how to create the device file if needed, possibly allocating an unused minor number. "md" causes a non-partitionable array to be used (though since Linux 2.6.28, these array devices are in fact partitionable). "mdp", "part" or "p" causes a partitionable array (2.6 and later) to be used. "yes" requires the named md device to have a 'standard' format, and the type and minor number will be determined from this. With *mdadm* 3.0, device creation is normally left up to *udev* so this option is unlikely to be needed. See DEVICE NAMES below.

The argument can also come immediately after "**-a**". e.g. "**-ap**".

If **--auto** is not given on the command line or in the config file, then the default will be **--auto=yes**.

If **--scan** is also given, then any *auto=* entries in the config file will override the **--auto** instruction given on the command line.

For partitionable arrays, *mdadm* will create the device file for the whole array and for the first 4 partitions. A different number of partitions can be specified at the end of this option (e.g. **--auto=p7**). If the device name ends with a digit, the partition names add a 'p', and a number, e.g. */dev/md/home1p3*. If there is no trailing digit, then the partition names just have a number added, e.g. */dev/md/scratch3*.

If the md device name is in a 'standard' format as described in DEVICE NAMES, then it will be created, if necessary, with the appropriate device number based on that name. If the device name is not in one of these formats, then a unused device number will be allocated. The device number will be considered unused if there is no active array for that number, and there is no entry in /dev for that number and with a non-standard name. Names that are not in 'standard' format are only allowed in "/dev/md/".

This is meaningful with **--create** or **--build**.

-a, --add

This option can be used in Grow mode in two cases.

If the target array is a Linear array, then **--add** can be used to add one or more devices to the array. They are simply catenated on to the end of the array. Once added, the devices cannot be removed.

If the **--raid-disks** option is being used to increase the number of devices in an array, then **--add** can be used to add some extra devices to be included in the array. In most cases this is not needed as the extra devices can be added as spares first, and then the number of raid-disks can be changed. However for RAID0, it is not possible to add spares. So to increase the number of devices in a RAID0, it is necessary to set the new number of devices, and to add the new devices, in the same command.

--nodes

Only works when the array is for clustered environment. It specifies the maximum number of nodes in the cluster that will use this device simultaneously. If not specified, this defaults to 4.

--write-journal

Specify journal device for the RAID-4/5/6 array. The journal device should be a SSD with reasonable lifetime.

--symlinks

Auto creation of symlinks in /dev to /dev/md, option **--symlinks** must be 'no' or 'yes' and work with **--create** and **--build**.

-k, --consistency-policy=

Specify how the array maintains consistency in case of unexpected shutdown. Only relevant for RAID levels with redundancy. Currently supported options are:

resync Full resync is performed and all redundancy is regenerated when the array is started after unclean shutdown.

bitmap Resync assisted by a write-intent bitmap. Implicitly selected when using **--bitmap**.

journal

For RAID levels 4/5/6, journal device is used to log transactions and replay after unclean shutdown. Implicitly selected when using **--write-journal**.

ppl For RAID5 only, Partial Parity Log is used to close the write hole and eliminate resync. PPL is stored in the metadata region of RAID member drives, no additional journal drive is needed.

Can be used with **--grow** to change the consistency policy of an active array in some cases. See CONSISTENCY POLICY CHANGES below.

For assemble:

-u, --uuid=

uuid of array to assemble. Devices which don't have this uuid are excluded

-m, --super-minor=

Minor number of device that array was created for. Devices which don't have this minor number are excluded. If you create an array as /dev/md1, then all superblocks will contain the minor number 1, even if the array is later assembled as /dev/md2.

Giving the literal word "dev" for **--super-minor** will cause *mdadm* to use the minor number of the md device that is being assembled. e.g. when assembling **/dev/md0, --super-minor=dev** will look for super blocks with a minor number of 0.

--super-minor is only relevant for v0.90 metadata, and should not normally be used. Using **--uuid** is much safer.

-N, --name=

Specify the name of the array to assemble. This must be the name that was specified when creating the array. It must either match the name stored in the superblock exactly, or it must match with the current *hostname* prefixed to the start of the given name.

-f, --force

Assemble the array even if the metadata on some devices appears to be out-of-date. If *mdadm* cannot find enough working devices to start the array, but can find some devices that are recorded as having failed, then it will mark those devices as working so that the array can be started. This works only for native. For external metadata it allows to start dirty degraded RAID 4, 5, 6. An array which requires **--force** to be started may contain data corruption. Use it carefully.

-R, --run

Attempt to start the array even if fewer drives were given than were present last time the array was active. Normally if not all the expected drives are found and **--scan** is not used, then the array will be assembled but not started. With **--run** an attempt will be made to start it anyway.

--no-degraded

This is the reverse of **--run** in that it inhibits the startup of array unless all expected drives are present. This is only needed with **--scan**, and can be used if the physical connections to devices are not as reliable as you would like.

-a, --auto{=no,yes,md,mdp,part}

See this option under Create and Build options.

-b, --bitmap=

Specify the bitmap file that was given when the array was created. If an array has an **internal** bitmap, there is no need to specify this when assembling the array.

--backup-file=

If **--backup-file** was used while reshaping an array (e.g. changing number of devices or chunk size) and the system crashed during the critical section, then the same **--backup-file** must be presented to **--assemble** to allow possibly corrupted data to be restored, and the reshape to be completed.

--invalid-backup

If the file needed for the above option is not available for any reason an empty file can be given together with this option to indicate that the backup file is invalid. In this case the data that was being rearranged at the time of the crash could be irrecoverably lost, but the rest of the array may still be recoverable. This option should only be used as a last resort if there is no way to recover the backup file.

-U, --update=

Update the superblock on each device while assembling the array. The argument given to this flag can be one of **sparc2.2**, **summaries**, **uuid**, **name**, **nodes**, **homehost**, **home-cluster**, **resync**, **byte-order**, **devicesize**, **no-bitmap**, **bbl**, **no-bbl**, **ppl**, **no-ppl**, **layout-original**, **layout-alternate**, **layout-unspecified**, **metadata**, or **super-minor**.

The **sparc2.2** option will adjust the superblock of an array what was created on a Sparc machine running a patched 2.2 Linux kernel. This kernel got the alignment of part of the superblock wrong. You can use the **--examine --sparc2.2** option to *mdadm* to see what effect this would have.

The **super-minor** option will update the **preferred minor** field on each superblock to match the minor number of the array being assembled. This can be useful if **--examine** reports a different "Preferred Minor" to **--detail**. In some cases this update will be performed automatically by the kernel driver. In particular the update happens automatically at the first write to an array with redundancy (RAID level 1 or greater) on a 2.6 (or later) kernel.

The **uuid** option will change the uuid of the array. If a UUID is given with the **--uuid** option that UUID will be used as a new UUID and will **NOT** be used to help identify the devices in the array. If no **--uuid** is given, a random UUID is chosen.

The **name** option will change the *name* of the array as stored in the superblock. This is only supported for version-1 superblocks.

The **nodes** option will change the *nodes* of the array as stored in the bitmap superblock. This option only works for a clustered environment.

The **homehost** option will change the *homehost* as recorded in the superblock. For version-0 superblocks, this is the same as updating the UUID. For version-1 superblocks, this involves updating the name.

The **home-cluster** option will change the cluster name as recorded in the superblock and bitmap. This option only works for clustered environment.

The **resync** option will cause the array to be marked *dirty* meaning that any redundancy in the array (e.g. parity for RAID5, copies for RAID1) may be incorrect. This will cause the RAID system to perform a "resync" pass to make sure that all redundant information is correct.

The **byteorder** option allows arrays to be moved between machines with different byte-order, such as from a big-endian machine like a Sparc or some MIPS machines, to a little-endian x86_64 machine. When assembling such an array for the first time after a move, giving **--update=byteorder** will cause *mdadm* to expect superblocks to have their byteorder reversed, and will correct that order before assembling the array. This is only valid with original (Version 0.90) superblocks.

The **summaries** option will correct the summaries in the superblock. That is the counts of total, working, active, failed, and spare devices.

The **devicesize** option will rarely be of use. It applies to version 1.1 and 1.2 metadata only (where the metadata is at the start of the device) and is only useful when the component device has changed size (typically become larger). The version 1 metadata records the amount of the device that can be used to store data, so if a device in a version 1.1 or 1.2 array becomes larger, the metadata will still be visible, but the extra space will not. In this case it might be useful to assemble the array with **--update=devicesize**. This will cause *mdadm* to determine the maximum usable amount of space on each device and update the relevant field in the metadata.

The **metadata** option only works on v0.90 metadata arrays and will convert them to v1.0 metadata. The array must not be dirty (i.e. it must not need a sync) and it must not have a write-intent bitmap.

The old metadata will remain on the devices, but will appear older than the new metadata and so will usually be ignored. The old metadata (or indeed the new metadata) can be removed by giving the appropriate **--metadata=** option to **--zero-superblock**.

The **no-bitmap** option can be used when an array has an internal bitmap which is corrupt in some way so that assembling the array normally fails. It will cause any internal bitmap to be ignored.

The **bbl** option will reserve space in each device for a bad block list. This will be 4K in size and positioned near the end of any free space between the superblock and the data.

The **no-bbl** option will cause any reservation of space for a bad block list to be removed. If the bad block list contains entries, this will fail, as removing the list could cause data corruption.

The **ppl** option will enable PPL for a RAID5 array and reserve space for PPL on each device. There must be enough free space between the data and superblock and a write-intent bitmap or journal must not be used.

The **no-ppl** option will disable PPL in the superblock.

The **layout-original** and **layout-alternate** options are for RAID0 arrays with non-uniform devices size that were in use before Linux 5.4. If the array was being used with Linux 3.13 or earlier, then to assemble the array on a new kernel, **--update=layout-original** must be given. If the

array was created and used with a kernel from Linux 3.14 to Linux 5.3, then **--update=layout-alternate** must be given. This only needs to be given once. Subsequent assembly of the array will happen normally. For more information, see *md(4)*.

The **layout-unspecified** option reverts the effect of **layout-original** or **layout-alternate** and allows the array to be again used on a kernel prior to Linux 5.3. This option should be used with great caution.

--freeze-reshape

Option is intended to be used in start-up scripts during initrd boot phase. When array under reshape is assembled during initrd phase, this option stops reshape after reshape critical section is being restored. This happens before file system pivot operation and avoids loss of file system context. Losing file system context would cause reshape to be broken.

Reshape can be continued later using the **--continue** option for the grow command.

--symlinks

See this option under Create and Build options.

For Manage mode:

-t, --test

Unless a more serious error occurred, *mdadm* will exit with a status of 2 if no changes were made to the array and 0 if at least one change was made. This can be useful when an indirect specifier such as **missing**, **detached** or **faulty** is used in requesting an operation on the array. **--test** will report failure if these specifiers didn't find any match.

-a, --add

hot-add listed devices. If a device appears to have recently been part of the array (possibly it failed or was removed) the device is re-added as described in the next point. If that fails or the device was never part of the array, the device is added as a hot-spare. If the array is degraded, it will immediately start to rebuild data onto that spare.

Note that this and the following options are only meaningful on array with redundancy. They don't apply to RAID0 or Linear.

--re-add

re-add a device that was previously removed from an array. If the metadata on the device reports that it is a member of the array, and the slot that it used is still vacant, then the device will be added back to the array in the same position. This will normally cause the data for that device to be recovered. However based on the event count on the device, the recovery may only require sections that are flagged by a write-intent bitmap to be recovered or may not require any recovery at all.

When used on an array that has no metadata (i.e. it was built with **--build**) it will be assumed that bitmap-based recovery is enough to make the device fully consistent with the array.

When used with v1.x metadata, **--re-add** can be accompanied by **--update=devicesize**, **--update=bbl**, or **--update=no-bbl**. See the description of these option when used in Assemble mode for an explanation of their use.

If the device name given is **missing** then *mdadm* will try to find any device that looks like it should be part of the array but isn't and will try to re-add all such devices.

If the device name given is **faulty** then *mdadm* will find all devices in the array that are marked **faulty**, remove them and attempt to immediately re-add them. This can be useful if you are certain that the reason for failure has been resolved.

--add--spare

Add a device as a spare. This is similar to **--add** except that it does not attempt **--re-add** first. The device will be added as a spare even if it looks like it could be a recent member of the array.

-r, --remove

remove listed devices. They must not be active. i.e. they should be failed or spare devices.

As well as the name of a device file (e.g. **/dev/v/sda1**) the words **failed**, **detached** and names like **set-A** can be given to **--remove**. The first causes all failed device to be removed. The second causes any device which is no longer connected to the system (i.e an 'open' returns **ENXIO**) to be removed. The third will remove a set as described below under **--fail**.

-f, --fail

Mark listed devices as faulty. As well as the name of a device file, the word **detached** or a set name like **set-A** can be given. The former will cause any device that has been detached from the system to be marked as failed. It can then be removed.

For RAID10 arrays where the number of copies evenly divides the number of devices, the devices can be conceptually divided into sets where each set contains a single complete copy of the data on the array. Sometimes a RAID10 array will be configured so that these sets are on separate controllers. In this case all the devices in one set can be failed by giving a name like **set-A** or **set-B** to **--fail**. The appropriate set names are reported by **--detail**.

--set-faulty

same as **--fail**.

--replace

Mark listed devices as requiring replacement. As soon as a spare is available, it will be rebuilt and will replace the marked device. This is similar to marking a device as faulty, but the device remains in service during the recovery process to increase resilience against multiple failures. When the replacement process finishes, the replaced device will be marked as faulty.

--with This can follow a list of **--replace** devices. The devices listed after **--with** will be preferentially used to replace the devices listed after **--replace**. These devices must already be spare devices in the array.

--write-mostly

Subsequent devices that are added or re-added will have the 'write-mostly' flag set. This is only valid for RAID1 and means that the 'md' driver will avoid reading from these devices if possible.

--readwrite

Subsequent devices that are added or re-added will have the 'write-mostly' flag cleared.

--cluster-confirm

Confirm the existence of the device. This is issued in response to an **--add** request by a node in a cluster. When a node adds a device it sends a message to all nodes in the cluster to look for a device with a UUID. This translates to a udev notification with the UUID of the device to be added and the slot number. The receiving node must acknowledge this message with **--cluster-confirm**. Valid arguments are <slot>:<devicename> in case the device is found or <slot>:missing in case the

device is not found.

--add-journal

Add journal to an existing array, or recreate journal for RAID-4/5/6 array that lost a journal device. To avoid interrupting on-going write operations, **--add-journal** only works for array in Read-Only state.

--failfast

Subsequent devices that are added or re-added will have the 'failfast' flag set. This is only valid for RAID1 and RAID10 and means that the 'md' driver will avoid long timeouts on error handling where possible.

--nofailfast

Subsequent devices that are re-added will be re-added without the 'failfast' flag set.

Each of these options requires that the first device listed is the array to be acted upon, and the remainder are component devices to be added, removed, marked as faulty, etc. Several different operations can be specified for different devices, e.g.

```
mdadm /dev/md0 --add /dev/sda1 --fail /dev/sdb1 --remove /dev/sdb1
```

Each operation applies to all devices listed until the next operation.

If an array is using a write-intent bitmap, then devices which have been removed can be re-added in a way that avoids a full reconstruction but instead just updates the blocks that have changed since the device was removed. For arrays with persistent metadata (superblocks) this is done automatically. For arrays created with **--build** mdadm needs to be told that this device was removed recently with **--re-add**.

Devices can only be removed from an array if they are not in active use, i.e. that must be spares or failed devices. To remove an active device, it must first be marked as **faulty**.

For Misc mode:

-Q, --query

Examine a device to see (1) if it is an md device and (2) if it is a component of an md array. Information about what is discovered is presented.

-D, --detail

Print details of one or more md devices.

--detail-platform

Print details of the platform's RAID capabilities (firmware / hardware topology) for a given metadata format. If used without argument, mdadm will scan all controllers looking for their capabilities. Otherwise, mdadm will only look at the controller specified by the argument in form of an absolute filepath or a link, e.g. */sys/devices/pci0000:00/0000:00:If.2*.

-Y, --export

When used with **--detail**, **--detail-platform**, **--examine**, or **--incremental** output will be formatted as **key=value** pairs for easy import into the environment.

With **--incremental** The value **MD_STARTED** indicates whether an array was started (**yes**) or not, which may include a reason (**unsafe**, **nothing**, **no**). Also the value **MD_FOREIGN** indicates if the array is expected on this host (**no**), or seems to be from elsewhere (**yes**).

-E, --examine

Print contents of the metadata stored on the named device(s). Note the contrast between **--examine** and **--detail**. **--examine** applies to devices which are components of an array, while **--detail** applies to a whole array which is currently active.

--sparc2.2

If an array was created on a SPARC machine with a 2.2 Linux kernel patched with RAID support, the superblock will have been created incorrectly, or at least incompatibly with 2.4 and later kernels. Using the **--sparc2.2** flag with **--examine** will fix the superblock before displaying it. If this appears to do the right thing, then the array can be successfully assembled using **--assemble --update=sparc2.2**.

-X, --examine-bitmap

Report information about a bitmap file. The argument is either an external bitmap file or an array component in case of an internal bitmap. Note that running this on an array device (e.g. **/dev/md0**) does not report the bitmap for that array.

--examine-badblocks

List the bad-blocks recorded for the device, if a bad-blocks list has been configured. Currently only **1.x** and **IMSM** metadata support bad-blocks lists.

--dump=directory**--restore=directory**

Save metadata from listed devices, or restore metadata to listed devices.

-R, --run

start a partially assembled array. If **--assemble** did not find enough devices to fully start the array, it might leave it partially assembled. If you wish, you can then use **--run** to start the array in degraded mode.

-S, --stop

deactivate array, releasing all resources.

-o, --readonly

mark array as readonly.

-w, --readwrite

mark array as readwrite.

--zero-superblock

If the device contains a valid md superblock, the block is overwritten with zeros. With **--force** the block where the superblock would be is overwritten even if it doesn't appear to be valid.

Note: Be careful to call **--zero-superblock** with clustered raid, make sure array isn't used or assembled in other cluster node before execute it.

--kill-subarray=

If the device is a container and the argument to **--kill-subarray** specifies an inactive subarray in the container, then the subarray is deleted. Deleting all subarrays will leave an 'empty-container' or spare superblock on the drives. See **--zero-superblock** for completely removing a superblock. Note that some formats depend on the subarray index for generating a UUID, this

command will fail if it would change the UUID of an active subarray.

--update–subarray=

If the device is a container and the argument to **--update–subarray** specifies a subarray in the container, then attempt to update the given superblock field in the subarray. See below in **MISC MODE** for details.

-t, --test

When used with **--detail**, the exit status of *mdadm* is set to reflect the status of the device. See below in **MISC MODE** for details.

-W, --wait

For each md device given, wait for any resync, recovery, or reshape activity to finish before returning. *mdadm* will return with success if it actually waited for every device listed, otherwise it will return failure.

--wait–clean

For each md device given, or each device in /proc/mdstat if **--scan** is given, arrange for the array to be marked clean as soon as possible. *mdadm* will return with success if the array uses external metadata and we successfully waited. For native arrays this returns immediately as the kernel handles dirty-clean transitions at shutdown. No action is taken if safe-mode handling is disabled.

--action=

Set the "sync_action" for all md devices given to one of **idle**, **frozen**, **check**, **repair**. Setting to **idle** will abort any currently running action though some actions will automatically restart. Setting to **frozen** will abort any current action and ensure no other action starts automatically.

Details of **check** and **repair** can be found in *md(4)* under **SCRUBBING AND MISMATCHES**.

For Incremental Assembly mode:

--rebuild–map, -r

Rebuild the map file (**/run/mdadm/map**) that *mdadm* uses to help track which arrays are currently being assembled.

--run, -R

Run any array assembled as soon as a minimal number of devices are available, rather than waiting until all expected devices are present.

--scan, -s

Only meaningful with **-R** this will scan the **map** file for arrays that are being incrementally assembled and will try to start any that are not already started. If any such array is listed in **mdadm.conf** as requiring an external bitmap, that bitmap will be attached first.

--fail, -f

This allows the hot-plug system to remove devices that have fully disappeared from the kernel. It will first fail and then remove the device from any array it belongs to. The device name given should be a kernel device name such as "sda", not a name in **/dev**.

--path=

Only used with --fail. The 'path' given will be recorded so that if a new device appears at the same location it can be automatically added to the same array. This allows the failed device to be automatically replaced by a new device without metadata if it appears at specified path. This option is normally only set by a *udev* script.

For Monitor mode:**-m, --mail**

Give a mail address to send alerts to.

-p, --program, --alert

Give a program to be run whenever an event is detected.

-y, --syslog

Cause all events to be reported through 'syslog'. The messages have facility of 'daemon' and varying priorities.

-d, --delay

Give a delay in seconds. *mdadm* polls the md arrays and then waits this many seconds before polling again. The default is 60 seconds. Since 2.6.16, there is no need to reduce this as the kernel alerts *mdadm* immediately when there is any change.

-r, --increment

Give a percentage increment. *mdadm* will generate RebuildNN events with the given percentage increment.

-f, --daemonic

Tell *mdadm* to run as a background daemon if it decides to monitor anything. This causes it to fork and run in the child, and to disconnect from the terminal. The process id of the child is written to stdout. This is useful with --scan which will only continue monitoring if a mail address or alert program is found in the config file.

-i, --pid-file

When *mdadm* is running in daemon mode, write the pid of the daemon process to the specified file, instead of printing it on standard output.

-1, --oneshot

Check arrays only once. This will generate **NewArray** events and more significantly **DegradedArray** and **SparesMissing** events. Running

mdadm --monitor --scan -1

from a cron script will ensure regular notification of any degraded arrays.

-t, --test

Generate a **TestMessage** alert for every array found at startup. This alert gets mailed and passed to the alert program. This can be used for testing that alert message do get through successfully.

--no-sharing

This inhibits the functionality for moving spares between arrays. Only one monitoring process started with --scan but without this flag is allowed, otherwise the two could interfere with each other.

ASSEMBLE MODE

Usage: **mdadm** **--assemble** *md-device options-and-component-devices...*

Usage: **mdadm** **--assemble** **--scan** *md-devices-and-options...*

Usage: **mdadm** **--assemble** **--scan** *options...*

This usage assembles one or more RAID arrays from pre-existing components. For each array, mdadm needs to know the md device, the identity of the array, and a number of component-devices. These can be found in a number of ways.

In the first usage example (without the **--scan**) the first device given is the md device. In the second usage example, all devices listed are treated as md devices and assembly is attempted. In the third (where no devices are listed) all md devices that are listed in the configuration file are assembled. If no arrays are described by the configuration file, then any arrays that can be found on unused devices will be assembled.

If precisely one device is listed, but **--scan** is not given, then *mdadm* acts as though **--scan** was given and identity information is extracted from the configuration file.

The identity can be given with the **--uuid** option, the **--name** option, or the **--super-minor** option, will be taken from the md-device record in the config file, or will be taken from the super block of the first component-device listed on the command line.

Devices can be given on the **--assemble** command line or in the config file. Only devices which have an md superblock which contains the right identity will be considered for any array.

The config file is only used if explicitly named with **--config** or requested with (a possibly implicit) **--scan**. In the later case, **/etc/mdadm/mdadm.conf** or **/etc/mdadm.conf** is used.

If **--scan** is not given, then the config file will only be used to find the identity of md arrays.

Normally the array will be started after it is assembled. However if **--scan** is not given and not all expected drives were listed, then the array is not started (to guard against usage errors). To insist that the array be started in this case (as may work for RAID1, 4, 5, 6, or 10), give the **--run** flag.

If *udev* is active, *mdadm* does not create any entries in **/dev** but leaves that to *udev*. It does record information in **/run/mdadm/map** which will allow *udev* to choose the correct name.

If *mdadm* detects that *udev* is not configured, it will create the devices in **/dev** itself.

In Linux kernels prior to version 2.6.28 there were two distinctly different types of md devices that could be created: one that could be partitioned using standard partitioning tools and one that could not. Since 2.6.28 that distinction is no longer relevant as both type of devices can be partitioned. *mdadm* will normally create the type that originally could not be partitioned as it has a well defined major number (9).

Prior to 2.6.28, it is important that *mdadm* chooses the correct type of array device to use. This can be controlled with the **--auto** option. In particular, a value of "mdp" or "part" or "p" tells *mdadm* to use a partitionable device rather than the default.

In the no-*udev* case, the value given to **--auto** can be suffixed by a number. This tells *mdadm* to create that number of partition devices rather than the default of 4.

The value given to **--auto** can also be given in the configuration file as a word starting **auto=** on the ARRAY line for the relevant array.

Auto Assembly

When **--assemble** is used with **--scan** and no devices are listed, *mdadm* will first attempt to assemble all the arrays listed in the config file.

If no arrays are listed in the config (other than those marked <ignore>) it will look through the available devices for possible arrays and will try to assemble anything that it finds. Arrays which are tagged as belonging to the given homehost will be assembled and started normally. Arrays which do not obviously belong to this host are given names that are expected not to conflict with anything local, and are started "read-auto" so that nothing is written to any device until the array is written to. i.e. automatic resync etc is delayed.

If *mdadm* finds a consistent set of devices that look like they should comprise an array, and if the superblock is tagged as belonging to the given home host, it will automatically choose a device name and try to assemble the array. If the array uses version-0.90 metadata, then the **minor** number as recorded in the superblock is used to create a name in **/dev/md/** so for example **/dev/md/3**. If the array uses version-1 metadata, then the **name** from the superblock is used to similarly create a name in **/dev/md/** (the name will have any 'host' prefix stripped first).

This behaviour can be modified by the **AUTO** line in the *mdadm.conf* configuration file. This line can indicate that specific metadata type should, or should not, be automatically assembled. If an array is found which is not listed in *mdadm.conf* and has a metadata format that is denied by the **AUTO** line, then it will not be assembled. The **AUTO** line can also request that all arrays identified as being for this homehost should be assembled regardless of their metadata type. See *mdadm.conf*(5) for further details.

Note: Auto assembly cannot be used for assembling and activating some arrays which are undergoing reshape. In particular as the**backup-file** cannot be given, any reshape which requires a backup-file to continue cannot be started by auto assembly. An array which is growing to more devices and has passed the critical section can be assembled using auto-assembly.

BUILD MODE

Usage: **mdadm --build** *md-device* **--chunk=X** **--level=Y** **--raid-devices=Z** *devices*

This usage is similar to **--create**. The difference is that it creates an array without a superblock. With these arrays there is no difference between initially creating the array and subsequently assembling the array, except that hopefully there is useful data there in the second case.

The level may raid0, linear, raid1, raid10, multipath, or faulty, or one of their synonyms. All devices must be listed and the array will be started once complete. It will often be appropriate to use **--assume-clean** with levels raid1 or raid10.

CREATE MODE

Usage: **mdadm --create** *md-device* **--chunk=X** **--level=Y**
--raid-devices=Z *devices*

This usage will initialise a new md array, associate some devices with it, and activate the array.

The named device will normally not exist when *mdadm --create* is run, but will be created by *udev* once the array becomes active.

The max length md-device name is limited to 32 characters. Different metadata types have more strict limitation (like IMSM where only 16 characters are allowed). For that reason, long name could be truncated or rejected, it depends on metadata policy.

As devices are added, they are checked to see if they contain RAID superblocks or filesystems. They are

also checked to see if the variance in device size exceeds 1%.

If any discrepancy is found, the array will not automatically be run, though the presence of a **--run** can override this caution.

To create a "degraded" array in which some devices are missing, simply give the word "**missing**" in place of a device name. This will cause *mdadm* to leave the corresponding slot in the array empty. For a RAID4 or RAID5 array at most one slot can be "**missing**"; for a RAID6 array at most two slots. For a RAID1 array, only one real device needs to be given. All of the others can be "**missing**".

When creating a RAID5 array, *mdadm* will automatically create a degraded array with an extra spare drive. This is because building the spare into a degraded array is in general faster than resyncing the parity on a non-degraded, but not clean, array. This feature can be overridden with the **--force** option.

When creating an array with version-1 metadata a name for the array is required. If this is not given with the **--name** option, *mdadm* will choose a name based on the last component of the name of the device being created. So if **/dev/md3** is being created, then the name **3** will be chosen. If **/dev/md/home** is being created, then the name **home** will be used.

When creating a partition based array, using *mdadm* with version-1.x metadata, the partition type should be set to **0xDA** (non fs-data). This type selection allows for greater precision since using any other [RAID auto-detect (0xFD) or a GNU/Linux partition (0x83)], might create problems in the event of array recovery through a live cdrom.

A new array will normally get a randomly assigned 128bit UUID which is very likely to be unique. If you have a specific need, you can choose a UUID for the array by giving the **--uuid=** option. Be warned that creating two arrays with the same UUID is a recipe for disaster. Also, using **--uuid=** when creating a v0.90 array will silently override any **--homehost=** setting.

If the array type supports a write-intent bitmap, and if the devices in the array exceed 100G in size, an internal write-intent bitmap will automatically be added unless some other option is explicitly requested with the **--bitmap** option or a different consistency policy is selected with the **--consistency-policy** option. In any case space for a bitmap will be reserved so that one can be added later with **--grow --bitmap=internal**.

If the metadata type supports it (currently only 1.x and IMSM metadata), space will be allocated to store a bad block list. This allows a modest number of bad blocks to be recorded, allowing the drive to remain in service while only partially functional.

When creating an array within a **CONTAINER** *mdadm* can be given either the list of devices to use, or simply the name of the container. The former case gives control over which devices in the container will be used for the array. The latter case allows *mdadm* to automatically choose which devices to use based on how much spare space is available.

The General Management options that are valid with **--create** are:

--run insist on running the array even if some devices look like they might be in use.

--readonly

start the array in readonly mode.

MANAGE MODE

Usage: **mdadm device options... devices...**

This usage will allow individual devices in an array to be failed, removed or added. It is possible to

perform multiple operations with one command. For example:

mdadm /dev/md0 -f /dev/hda1 -r /dev/hda1 -a /dev/hda1

will firstly mark **/dev/hda1** as faulty in **/dev/md0** and will then remove it from the array and finally add it back in as a spare. However only one md array can be affected by a single command.

When a device is added to an active array, mdadm checks to see if it has metadata on it which suggests that it was recently a member of the array. If it does, it tries to "re-add" the device. If there have been no changes since the device was removed, or if the array has a write-intent bitmap which has recorded whatever changes there were, then the device will immediately become a full member of the array and those differences recorded in the bitmap will be resolved.

MISC MODE

Usage: **mdadm options ... devices ...**

MISC mode includes a number of distinct operations that operate on distinct devices. The operations are:

--query

The device is examined to see if it is (1) an active md array, or (2) a component of an md array.
The information discovered is reported.

--detail

The device should be an active md device. **mdadm** will display a detailed description of the array. **--brief** or **--scan** will cause the output to be less detailed and the format to be suitable for inclusion in **mdadm.conf**. The exit status of *mdadm* will normally be 0 unless *mdadm* failed to get useful information about the device(s); however, if the **--test** option is given, then the exit status will be:

- 0 The array is functioning normally.
- 1 The array has at least one failed device.
- 2 The array has multiple failed devices such that it is unusable.
- 4 There was an error while trying to get information about the device.

--detail-platform

Print detail of the platform's RAID capabilities (firmware / hardware topology). If the metadata is specified with **-e** or **--metadata=** then the return status will be:

- 0 metadata successfully enumerated its platform components on this system
- 1 metadata is platform independent
- 2 metadata failed to find its platform components on this system

--update-subarray=

If the device is a container and the argument to **--update-subarray** specifies a subarray in the container, then attempt to update the given superblock field in the subarray. Similar to updating an array in "assemble" mode, the field to update is selected by **-U** or **--update=** option. The supported options are **name**, **ppl**, **no-ppl**, **bitmap** and **no-bitmap**.

The **name** option updates the subarray name in the metadata, it may not affect the device node name or the device node symlink until the subarray is re-assembled. If updating **name** would change the UUID of an active subarray this operation is blocked, and the command will end in an error.

The **ppl** and **no-ppl** options enable and disable PPL in the metadata. Currently supported only for IMSM subarrays.

The **bitmap** and **no-bitmap** options enable and disable write-intent bitmap in the metadata. Currently supported only for IMSM subarrays.

--examine

The device should be a component of an md array. *mdadm* will read the md superblock of the device and display the contents. If **--brief** or **--scan** is given, then multiple devices that are components of the one array are grouped together and reported in a single entry suitable for inclusion in **mdadm.conf**.

Having **--scan** without listing any devices will cause all devices listed in the config file to be examined.

--dump=directory

If the device contains RAID metadata, a file will be created in the *directory* and the metadata will be written to it. The file will be the same size as the device and have the metadata written in the file at the same locate that it exists in the device. However the file will be "sparse" so that only those blocks containing metadata will be allocated. The total space used will be small.

The file name used in the *directory* will be the base name of the device. Further if any links appear in */dev/disk/by-id* which point to the device, then hard links to the file will be created in *directory* based on these *by-id* names.

Multiple devices can be listed and their metadata will all be stored in the one directory.

--restore=directory

This is the reverse of **--dump**. *mdadm* will locate a file in the directory that has a name appropriate for the given device and will restore metadata from it. Names that match */dev/disk/by-id* names are preferred, however if two of those refer to different files, *mdadm* will not choose between them but will abort the operation.

If a file name is given instead of a *directory* then *mdadm* will restore from that file to a single device, always provided the size of the file matches that of the device, and the file contains valid metadata.

--stop The devices should be active md arrays which will be deactivated, as long as they are not currently in use.

--run This will fully activate a partially assembled md array.

--readonly

This will mark an active array as read-only, providing that it is not currently being used.

--readwrite

This will change a **readonly** array back to being read/write.

--scan For all operations except **--examine**, **--scan** will cause the operation to be applied to all arrays listed in */proc/mdstat*. For **--examine**, **--scan** causes all devices listed in the config file to be examined.

-b, --brief

Be less verbose. This is used with **--detail** and **--examine**. Using **--brief** with **--verbose** gives an intermediate level of verbosity.

MONITOR MODE

Usage: **mdadm --monitor** *options... devices...*

This usage causes *mdadm* to periodically poll a number of md arrays and to report on any events noticed. *mdadm* will never exit once it decides that there are arrays to be checked, so it should normally be run in the background.

As well as reporting events, *mdadm* may move a spare drive from one array to another if they are in the same **spare-group** or **domain** and if the destination array has a failed drive but no spares.

If any devices are listed on the command line, *mdadm* will only monitor those devices. Otherwise all arrays listed in the configuration file will be monitored. Further, if **--scan** is given, then any other md devices that appear in **/proc/mdstat** will also be monitored.

The result of monitoring the arrays is the generation of events. These events are passed to a separate program (if specified) and may be mailed to a given E-mail address.

When passing events to a program, the program is run once for each event, and is given 2 or 3 command-line arguments: the first is the name of the event (see below), the second is the name of the md device which is affected, and the third is the name of a related device if relevant (such as a component device that has failed).

If **--scan** is given, then a program or an E-mail address must be specified on the command line or in the config file. If neither are available, then *mdadm* will not monitor anything. Without **--scan**, *mdadm* will continue monitoring as long as something was found to monitor. If no program or email is given, then each event is reported to **stdout**.

The different events are:

DeviceDisappeared

An md array which previously was configured appears to no longer be configured. (syslog priority: Critical)

If *mdadm* was told to monitor an array which is RAID0 or Linear, then it will report **DeviceDisappeared** with the extra information **Wrong-Level**. This is because RAID0 and Linear do not support the device-failed, hot-spare and resync operations which are monitored.

RebuildStarted

An md array started reconstruction (e.g. recovery, resync, reshape, check, repair). (syslog priority: Warning)

RebuildNN

Where *NN* is a two-digit number (ie. 05, 48). This indicates that rebuild has passed that many percent of the total. The events are generated with fixed increment since 0. Increment size may be specified with a commandline option (default is 20). (syslog priority: Warning)

RebuildFinished

An md array that was rebuilding, isn't any more, either because it finished normally or was aborted. (syslog priority: Warning)

Fail

An active component device of an array has been marked as faulty. (syslog priority: Critical)

FailSpare

A spare component device which was being rebuilt to replace a faulty device has failed. (syslog priority: Critical)

SpareActive

A spare component device which was being rebuilt to replace a faulty device has been successfully rebuilt and has been made active. (syslog priority: Info)

NewArray

A new md array has been detected in the **/proc/mdstat** file. (syslog priority: Info)

DegradedArray

A newly noticed array appears to be degraded. This message is not generated when *mdadm* notices a drive failure which causes degradation, but only when *mdadm* notices that an array is degraded when it first sees the array. (syslog priority: Critical)

MoveSpare

A spare drive has been moved from one array in a **spare-group** or **domain** to another to allow a failed drive to be replaced. (syslog priority: Info)

SparesMissing

If *mdadm* has been told, via the config file, that an array should have a certain number of spare devices, and *mdadm* detects that it has fewer than this number when it first sees the array, it will report a **SparesMissing** message. (syslog priority: Warning)

TestMessage

An array was found at startup, and the **--test** flag was given. (syslog priority: Info)

Only **Fail**, **FailSpare**, **DegradedArray**, **SparesMissing** and **TestMessage** cause Email to be sent. All events cause the program to be run. The program is run with two or three arguments: the event name, the array device and possibly a second device.

Each event has an associated array device (e.g. **/de v/***md1*) and possibly a second device. For **Fail**, **FailSpare**, and **SpareActive** the second device is the relevant component device. For **MoveSpare** the second device is the array that the spare was moved from.

For *mdadm* to move spares from one array to another, the different arrays need to be labeled with the same **spare-group** or the spares must be allowed to migrate through matching POLICY domains in the configuration file. The **spare-group** name can be any string; it is only necessary that different spare groups use different names.

When *mdadm* detects that an array in a spare group has fewer active devices than necessary for the complete array, and has no spare devices, it will look for another array in the same spare group that has a full complement of working drive and a spare. It will then attempt to remove the spare from the second drive and add it to the first. If the removal succeeds but the adding fails, then it is added back to the original array.

If the spare group for a degraded array is not defined, *mdadm* will look at the rules of spare migration specified by POLICY lines in **mdadm.conf** and then follow similar steps as above if a matching spare is found.

GROW MODE

The GROW mode is used for changing the size or shape of an active array. For this to work, the kernel must support the necessary change. Various types of growth are being added during 2.6 development.

Currently the supported changes include

- change the "size" attribute for RAID1, RAID4, RAID5 and RAID6.
- increase or decrease the "raid-devices" attribute of RAID0, RAID1, RAID4, RAID5, and RAID6.
- change the chunk-size and layout of RAID0, RAID4, RAID5, RAID6 and RAID10.
- convert between RAID1 and RAID5, between RAID5 and RAID6, between RAID0, RAID4, and RAID5, and between RAID0 and RAID10 (in the near-2 mode).
- add a write-intent bitmap to any array which supports these bitmaps, or remove a write-intent bitmap from such an array.
- change the array's consistency policy.

Using GROW on containers is currently supported only for Intel's IMSM container format. The number of devices in a container can be increased - which affects all arrays in the container - or an array in a container can be converted between levels where those levels are supported by the container, and the conversion is one of those listed above.

Notes:

- Intel's native checkpointing doesn't use **--backup-file** option and it is transparent for assembly feature.
- Roaming between Windows(R) and Linux systems for IMSM metadata is not supported during grow process.
- When growing a raid0 device, the new component disk size (or external backup size) should be larger than $\text{LCM}(\text{old}, \text{new}) * \text{chunk-size} * 2$, where $\text{LCM}()$ is the least common multiple of the old and new count of component disks, and " $* 2$ " comes from the fact that mdadm refuses to use more than half of a spare device for backup space.

SIZE CHANGES

Normally when an array is built the "size" is taken from the smallest of the drives. If all the small drives in an array are, one at a time, removed and replaced with larger drives, then you could have an array of large drives with only a small amount used. In this situation, changing the "size" with "GROW" mode will allow the extra space to start being used. If the size is increased in this way, a "resync" process will start to make sure the new parts of the array are synchronised.

Note that when an array changes size, any filesystem that may be stored in the array will not automatically grow or shrink to use or vacate the space. The filesystem will need to be explicitly told to use the extra space after growing, or to reduce its size **prior** to shrinking the array.

Also the size of an array cannot be changed while it has an active bitmap. If an array has a bitmap, it must be removed before the size can be changed. Once the change is complete a new bitmap can be created.

Note: **--grow --size** is not yet supported for external file bitmap.

RAID-DEVICES CHANGES

A RAID1 array can work with any number of devices from 1 upwards (though 1 is not very useful). There may be times which you want to increase or decrease the number of active devices. Note that this is different to hot-add or hot-remove which changes the number of inactive devices.

When reducing the number of devices in a RAID1 array, the slots which are to be removed from the array

must already be vacant. That is, the devices which were in those slots must be failed and removed.

When the number of devices is increased, any hot spares that are present will be activated immediately.

Changing the number of active devices in a RAID5 or RAID6 is much more effort. Every block in the array will need to be read and written back to a new location. From 2.6.17, the Linux Kernel is able to increase the number of devices in a RAID5 safely, including restarting an interrupted "reshape". From 2.6.31, the Linux Kernel is able to increase or decrease the number of devices in a RAID5 or RAID6.

From 2.6.35, the Linux Kernel is able to convert a RAID0 in to a RAID4 or RAID5. *mdadm* uses this functionality and the ability to add devices to a RAID4 to allow devices to be added to a RAID0. When requested to do this, *mdadm* will convert the RAID0 to a RAID4, add the necessary disks and make the reshape happen, and then convert the RAID4 back to RAID0.

When decreasing the number of devices, the size of the array will also decrease. If there was data in the array, it could get destroyed and this is not reversible, so you should firstly shrink the filesystem on the array to fit within the new size. To help prevent accidents, *mdadm* requires that the size of the array be decreased first with **mdadm --grow --array-size**. This is a reversible change which simply makes the end of the array inaccessible. The integrity of any data can then be checked before the non-reversible reduction in the number of devices is requested.

When relocating the first few stripes on a RAID5 or RAID6, it is not possible to keep the data on disk completely consistent and crash-proof. To provide the required safety, *mdadm* disables writes to the array while this "critical section" is reshaped, and takes a backup of the data that is in that section. For grows, this backup may be stored in any spare devices that the array has, however it can also be stored in a separate file specified with the **--backup-file** option, and is required to be specified for shrinks, RAID level changes and layout changes. If this option is used, and the system does crash during the critical period, the same file must be passed to **--assemble** to restore the backup and reassemble the array. When shrinking rather than growing the array, the reshape is done from the end towards the beginning, so the "critical section" is at the end of the reshape.

LEVEL CHANGES

Changing the RAID level of any array happens instantaneously. However in the RAID5 to RAID6 case this requires a non-standard layout of the RAID6 data, and in the RAID6 to RAID5 case that non-standard layout is required before the change can be accomplished. So while the level change is instant, the accompanying layout change can take quite a long time. A **--backup-file** is required. If the array is not simultaneously being grown or shrunk, so that the array size will remain the same - for example, reshaping a 3-drive RAID5 into a 4-drive RAID6 - the backup file will be used not just for a "critical section" but throughout the reshape operation, as described below under LAYOUT CHANGES.

CHUNK-SIZE AND LAYOUT CHANGES

Changing the chunk-size or layout without also changing the number of devices as the same time will involve re-writing all blocks in-place. To ensure against data loss in the case of a crash, a **--backup-file** must be provided for these changes. Small sections of the array will be copied to the backup file while they are being rearranged. This means that all the data is copied twice, once to the backup and once to the new layout on the array, so this type of reshape will go very slowly.

If the reshape is interrupted for any reason, this backup file must be made available to **mdadm --assemble** so the array can be reassembled. Consequently the file cannot be stored on the device being reshaped.

BITMAP CHANGES

A write-intent bitmap can be added to, or removed from, an active array. Either internal bitmaps, or bitmaps stored in a separate file, can be added. Note that if you add a bitmap stored in a file which is in a filesystem that is on the RAID array being affected, the system will deadlock. The bitmap must be on a separate filesystem.

CONSISTENCY POLICY CHANGES

The consistency policy of an active array can be changed by using the **--consistency-policy** option in Grow mode. Currently this works only for the **ppl** and **resync** policies and allows one to enable or disable the RAID5 Partial Parity Log (PPL).

INCREMENTAL MODE

Usage: **mdadm --incremental** [**--run**] [**--quiet**] *component-device* [*optional-aliases-for-device*]

Usage: **mdadm --incremental --fail** *component-device*

Usage: **mdadm --incremental --rebuild-map**

Usage: **mdadm --incremental --run --scan**

This mode is designed to be used in conjunction with a device discovery system. As devices are found in a system, they can be passed to **mdadm --incremental** to be conditionally added to an appropriate array.

Conversely, it can also be used with the **--fail** flag to do just the opposite and find whatever array a particular device is part of and remove the device from that array.

If the device passed is a **CONTAINER** device created by a previous call to *mdadm*, then rather than trying to add that device to an array, all the arrays described by the metadata of the container will be started.

mdadm performs a number of tests to determine if the device is part of an array, and which array it should be part of. If an appropriate array is found, or can be created, *mdadm* adds the device to the array and conditionally starts the array.

Note that *mdadm* will normally only add devices to an array which were previously working (active or spare) parts of that array. The support for automatic inclusion of a new drive as a spare in some array requires a configuration through POLICY in config file.

The tests that *mdadm* makes are as follow:

- + Is the device permitted by **mdadm.conf**? That is, is it listed in a**DEVICES** line in that file. If **DEVICES** is absent then the default it to allow any device. Similarly if**DEVICES** contains the special word **partitions** then any device is allowed. Otherwise the device name given to *mdadm*, or one of the aliases given, or an alias found in the filesystem, must match one of the names or patterns in a **DEVICES** line.

This is the only context where the aliases are used. They are usually provided by a *udev* rules mentioning **\$env{DEVLINKS}**.

- + Does the device have a valid md superblock? If a specific metadata version is requested with **--metadata** or **-e** then only that style of metadata is accepted, otherwise *mdadm* finds any known version of metadata. If no *md* metadata is found, the device may be still added to an array as a spare if POLICY allows.

mdadm keeps a list of arrays that it has partially assembled in **/run/mdadm/map**. If no array exists which

matches the metadata on the new device, *mdadm* must choose a device name and unit number. It does this based on any name given in **mdadm.conf** or any name information stored in the metadata. If this name suggests a unit number, that number will be used, otherwise a free unit number will be chosen. Normally *mdadm* will prefer to create a partitionable array, however if the **CREATE** line in **mdadm.conf** suggests that a non-partitionable array is preferred, that will be honoured.

If the array is not found in the config file and its metadata does not identify it as belonging to the "home-host", then *mdadm* will choose a name for the array which is certain not to conflict with any array which does belong to this host. It does this by adding an underscore and a small number to the name preferred by the metadata.

Once an appropriate array is found or created and the device is added, *mdadm* must decide if the array is ready to be started. It will normally compare the number of available (non-spare) devices to the number of devices that the metadata suggests need to be active. If there are at least that many, the array will be started. This means that if any devices are missing the array will not be restarted.

As an alternative, **--run** may be passed to *mdadm* in which case the array will be run as soon as there are enough devices present for the data to be accessible. For a RAID1, that means one device will start the array. For a clean RAID5, the array will be started as soon as all but one drive is present.

Note that neither of these approaches is really ideal. If it can be known that all device discovery has completed, then

mdadm -IRs

can be run which will try to start all arrays that are being incrementally assembled. They are started in "read-auto" mode in which they are read-only until the first write request. This means that no metadata updates are made and no attempt at resync or recovery happens. Further devices that are found before the first write can still be added safely.

ENVIRONMENT

This section describes environment variables that affect how *mdadm* operates.

MDADM_NO_MDMON

Setting this value to 1 will prevent *mdadm* from automatically launching *mdmon*. This variable is intended primarily for debugging *mdadm*/*mdmon*.

MDADM_NO_UDEV

Normally, *mdadm* does not create any device nodes in /dev, but leaves that task to *udev*. If *udev* v appears not to be configured, or if this environment variable is set to '1', the *mdadm* will create and devices that are needed.

MDADM_NO_SYSTEMCTL

If *mdadm* detects that *systemd* is in use it will normally request *systemd* to start various background tasks (particularly *mdmon*) rather than forking and running them in the background. This can be suppressed by setting **MDADM_NO_SYSTEMCTL=1**.

IMSM_NO_PLATFORM

A key value of IMSM metadata is that it allows interoperability with boot ROMs on Intel platforms, and with other major operating systems. Consequently, *mdadm* will only allow an IMSM array to be created or modified if detects that it is running on an Intel platform which supports IMSM, and supports the particular configuration of IMSM that is being requested (some functionality requires newer OROM support).

These checks can be suppressed by setting `IMSM_NO_PLATFORM=1` in the environment. This can be useful for testing or for disaster recovery. You should be aware that interoperability may be compromised by setting this value.

MDADM_GROW_ALLOW_OLD

If an array is stopped while it is performing a reshape and that reshape was making use of a backup file, then when the array is re-assembled `mdadm` will sometimes complain that the backup file is too old. If this happens and you are certain it is the right backup file, you can over-ride this check by setting `MDADM_GROW_ALLOW_OLD=1` in the environment.

MDADM_CONF_AUTO

Any string given in this variable is added to the start of the **AUTO** line in the config file, or treated as the whole **AUTO** line if none is given. It can be used to disable certain metadata types when `mdadm` is called from a boot script. For example

```
export MDADM_CONF_AUTO=-ddf -imsm
```

will make sure that `mdadm` does not automatically assemble any DDF or IMSM arrays that are found. This can be useful on systems configured to manage such arrays with **dmraid**.

EXAMPLES

mdadm --query /dev/name-of-device

This will find out if a given device is a RAID array, or is part of one, and will provide brief information about the device.

mdadm --assemble --scan

This will assemble and start all arrays listed in the standard config file. This command will typically go in a system startup file.

mdadm --stop --scan

This will shut down all arrays that can be shut down (i.e. are not currently in use). This will typically go in a system shutdown script.

mdadm --follow --scan --delay=120

If (and only if) there is an Email address or program given in the standard config file, then monitor the status of all arrays listed in that file by polling them ever 2 minutes.

mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/hd[ac]1

Create `/dev/md0` as a RAID1 array consisting of `/dev/hda1` and `/dev/hdc1`.

echo 'DEVICE /dev/hd*[0-9] /dev/sd*[0-9]' > mdadm.conf

mdadm --detail --scan >> mdadm.conf

This will create a prototype config file that describes currently active arrays that are known to be made from partitions of IDE or SCSI drives. This file should be reviewed before being used as it may contain unwanted detail.

echo 'DEVICE /dev/hd[a-z] /dev/sd*[a-z]' > mdadm.conf

mdadm --examine --scan --config=mdadm.conf >> mdadm.conf

This will find arrays which could be assembled from existing IDE and SCSI whole drives (not partitions), and store the information in the format of a config file. This file is very likely to contain unwanted detail, particularly the `devices=` entries. It should be reviewed and edited before being used as an actual config file.

mdadm --examine --brief --scan --config=partitions

mdadm -Ebsc partitions

Create a list of devices by reading **/proc/partitions**, scan these for RAID superblocks, and printout a brief listing of all that were found.

mdadm -Ac partitions -m 0 /dev/md0

Scan all partitions and devices listed in **/proc/partitions** and assemble **/dev/md0** out of all such devices with a RAID superblock with a minor number of 0.

mdadm --monitor --scan --daemonise > /run/mdadm/mon.pid

If config file contains a mail address or alert program, run mdadm in the background in monitor mode monitoring all md devices. Also write pid of mdadm daemon to **/run/mdadm/mon.pid**.

mdadm -Iq /dev/somedevice

Try to incorporate newly discovered device into some array as appropriate.

mdadm --incremental --rebuild-map --run --scan

Rebuild the array map from any current arrays, and then start any that can be started.

mdadm /dev/md4 --fail detached --remove detached

Any devices which are components of /dev/md4 will be marked as faulty and then remove from the array.

mdadm --grow /dev/md4 --level=6 --backup-file=/root/backup-md4

The array **/dev/md4** which is currently a RAID5 array will be converted to RAID6. There should normally already be a spare drive attached to the array as a RAID6 needs one more drive than a matching RAID5.

mdadm --create /dev/md/ddf --metadata=ddf --raid-disks 6 /dev/sd[a-f]

Create a DDF array over 6 devices.

mdadm --create /dev/md/home -n3 -l5 -z 30000000 /dev/md/ddf

Create a RAID5 array over any 3 devices in the given DDF set. Use only 30 gigabytes of each device.

mdadm -A /dev/md/ddf1 /dev/sd[a-f]

Assemble a pre-exist ddf array.

mdadm -I /dev/md/ddf1

Assemble all arrays contained in the ddf array, assigning names as appropriate.

mdadm --create --help

Provide help about the Create mode.

mdadm --config --help

Provide help about the format of the config file.

mdadm --help

Provide general help.

FILES

/proc/mdstat

If you're using the **/proc** filesystem, **/proc/mdstat** lists all active md devices with information about them. *mdadm* uses this to find arrays when **--scan** is given in Misc mode, and to monitor array reconstruction on Monitor mode.

/etc/mdadm/mdadm.conf (or /etc/mdadm.conf)

The config file lists which devices may be scanned to see if they contain MD super block, and gives identifying information (e.g. UUID) about known MD arrays. See **mdadm.conf(5)** for more details.

/etc/mdadm/mdadm.conf.d (or /etc/mdadm.conf.d)

A directory containing configuration files which are read in lexical order.

/run/mdadm/map

When **--incremental** mode is used, this file gets a list of arrays currently being created.

DEVICE NAMES

mdadm understand two sorts of names for array devices.

The first is the so-called 'standard' format name, which matches the names used by the kernel and which appear in */proc/mdstat*.

The second sort can be freely chosen, but must reside in */dev/md/*. When giving a device name to *mdadm* to create or assemble an array, either full path name such as */dev/md0* or */dev/md/home* can be given, or just the suffix of the second sort of name, such as *home* can be given.

When *mdadm* chooses device names during auto-assembly or incremental assembly, it will sometimes add a small sequence number to the end of the name to avoid conflicted between multiple arrays that have the same name. If *mdadm* can reasonably determine that the array really is meant for this host, either by a hostname in the metadata, or by the presence of the array in **mdadm.conf**, then it will leave off the suffix if possible. Also if the homehost is specified as<ignore> *mdadm* will only use a suffix if a different array of the same name already exists or is listed in the config file.

The standard names for non-partitioned arrays (the only sort of md array available in 2.4 and earlier) are of the form

/dev/mdNN

where NN is a number. The standard names for partitionable arrays (as available from 2.6 onwards) are of the form:

/dev/md_dNN

Partition numbers should be indicated by adding "pMM" to these, thus "*/dev/md/d1p2*".

From kernel version 2.6.28 the "non-partitioned array" can actually be partitioned. So the "md_dNN" names are no longer needed, and partitions such as "*/dev/mdNNpXX*" are possible.

From kernel version 2.6.29 standard names can be non-numeric following the form:

/dev/md_XXX

where **XXX** is any string. These names are supported by *mdadm* since version 3.3 provided they are enabled in *mdadm.conf*.

NOTE

mdadm was previously known as *mdctl*.

SEE ALSO

For further information on mdadm usage, MD and the various levels of RAID, see:

<https://raid.wiki.kernel.org/>

(based upon Jakob Østergaard's Software-RAID.HOWTO)

The latest version of *mdadm* should always be available from

<https://www.kernel.org/pub/linux/utils/raid/mdadm/>

Related man pages:

mdmon(8), *mdadm.conf(5)*, *md(4)*.

NAME

`mdadm.conf` – configuration for management of Software RAID with `mdadm`

SYNOPSIS

`/etc/mdadm/mdadm.conf`

DESCRIPTION

`mdadm` is a tool for creating, managing, and monitoring RAID devices using the **md** driver in Linux.

Some common tasks, such as assembling all arrays, can be simplified by describing the devices and arrays in this configuration file.

SYNTAX

The file should be seen as a collection of words separated by white space (space, tab, or newline). Any word that begins with a hash sign (#) starts a comment and that word together with the remainder of the line is ignored.

Spaces can be included in a word using quotation characters. Either single quotes ('') or double quotes ("") may be used. All the characters from one quotation character to next identical character are protected and will not be used to separate words to start new quoted strings. To include a single quote it must be between double quotes. To include a double quote it must be between single quotes.

Any line that starts with white space (space or tab) is treated as though it were a continuation of the previous line.

Empty lines are ignored, but otherwise each (non continuation) line must start with a keyword as listed below. The keywords are case insensitive and can be abbreviated to 3 characters.

The keywords are:

DEVICE

A **device** line lists the devices (whole devices or partitions) that might contain a component of an MD array. When looking for the components of an array, `mdadm` will scan these devices (or any devices listed on the command line).

The **device** line may contain a number of different devices (separated by spaces) and each device name can contain wild cards as defined by **glob(7)**.

Also, there may be several device lines present in the file.

Alternatively, a **device** line can contain either or both of the words **containers** and **partitions**. The word **containers** will cause `mdadm` to look for assembled CONTAINER arrays and include them as a source for assembling further arrays.

The word **partitions** will cause `mdadm` to read `/proc/partitions` and include all devices and partitions found therein. `mdadm` does not use the names from `/proc/partitions` but only the major and minor device numbers. It scans `/dev` to find the name that matches the numbers.

If no DEVICE line is present, then "DEVICE partitions containers" is assumed.

For example:

```
DEVICE /dev/hda* /dev/hdc*
DEV  /dev/sd*
DEVICE /dev/disk/by-path/pci*
DEVICE partitions
```

ARRAY

The ARRAY lines identify actual arrays. The second word on the line may be the name of the device where the array is normally assembled, such as **/dev/md1** or **/dev/md/backup**. If the name does not start with a slash ('/'), it is treated as being in **/dev/md/**. Alternately the word **<ignore>** (complete with angle brackets) can be given in which case any array which matches the rest of the line will never be automatically assembled. If no device name is given, *mdadm* will use various heuristics to determine an appropriate name.

Subsequent words identify the array, or identify the array as a member of a group. If multiple identities are given, then a component device must match ALL identities to be considered a match. Each identity word has a tag, and equals sign, and some value. The tags are:

uuid= The value should be a 128 bit uuid in hexadecimal, with punctuation interspersed if desired. This must match the uuid stored in the superblock.

name= The value should be a simple textual name as was given to *mdadm* when the array was created. This must match the name stored in the superblock on a device for that device to be included in the array. Not all superblock formats support names.

super-minor=

The value is an integer which indicates the minor number that was stored in the superblock when the array was created. When an array is created as **/dev/mdX**, then the minor number X is stored.

devices=

The value is a comma separated list of device names or device name patterns. Only devices with names which match one entry in the list will be used to assemble the array. Note that the devices listed there must also be listed on a DEVICE line.

level= The value is a RAID level. This is not normally used to identify an array, but is supported so that the output of

mdadm --examine --scan

can be used directly in the configuration file.

num-devices=

The value is the number of devices in a complete active array. As with **level** **vel=** this is mainly for compatibility with the output of

mdadm --examine --scan.

spares=

The value is a number of spare devices to expect the array to have. The sole use of this keyword and value is as follows: **mdadm --monitor** will report an array if it is found to have fewer than this number of spares when **--monitor** starts or when **--oneshot** is used.

spare-group=

The value is a textual name for a group of arrays. All arrays with the same **spare-group** name are considered to be part of the same group. The significance of a group of arrays is that *mdadm* will, when monitoring the arrays, move a spare drive from one array in a group to another array in that group if the first array had a failed or missing drive but no spare.

auto= This option is rarely needed with *mdadm*-3.0, particularly if use with the Linux kernel v2.6.28 or later. It tells *mdadm* whether to use partitionable array or non-partitionable arrays and, in the absence of *udev*, how many partition devices to create. From 2.6.28 all md array devices are partitionable, hence this option is not needed.

The value of this option can be "yes" or "md" to indicate that a traditional, non-partitionable md array should be created, or "mdp", "part" or "partition" to indicate that a partitionable md array (only available in linux 2.6 and later) should be used. This later set can also have a number appended to indicate how many partitions to create device files for, e.g. **auto=mdp5**. The default is 4.

bitmap=

The option specifies a file in which a write-intent bitmap should be found. When assembling the array, *mdadm* will provide this file to the **md** driver as the bitmap file. This has the same function as the **--bitmap-file** option to **--assemble**.

metadata=

Specify the metadata format that the array has. This is mainly recognised for comparability with the output of **mdadm -Es**.

container=

Specify that this array is a member array of some container. The value given can be either a path name in /dev, or a UUID of the container array.

member=

Specify that this array is a member array of some container. Each type of container has some way to enumerate member arrays, often a simple sequence number. The value identifies which member of a container the array is. It will usually accompany a "container=" word.

MAILADDR

The **mailaddr** line gives an E-mail address that alerts should be sent to when *mdadm* is running in **--monitor** mode (and was given the **--scan** option). There should only be one **MAILADDR** line and it should have only one address. Any subsequent addresses are silently ignored.

MAILFROM

The **mailfrom** line (which can only be abbreviated to at least 5 characters) gives an address to appear in the "From" address for alert mails. This can be useful if you want to explicitly set a domain, as the default from address is "root" with no domain. All words on this line are catenated with spaces to form the address.

Note that this value cannot be set via the *mdadm* commandline. It is only settable via the config file.

PROGRAM

The **program** line gives the name of a program to be run when **mdadm --monitor** detects potentially interesting events on any of the arrays that it is monitoring. This program gets run with two or three arguments, they being the Event, the md device, and possibly the related component device.

There should only be one **program** line and it should give only one program.

CREATE

The **create** line gives default values to be used when creating arrays, new members of arrays, and device entries for arrays. These include:

owner=**group=**

These can give user/group ids or names to use instead of system defaults (root/wheel or root/disk).

mode= An octal file mode such as 0660 can be given to override the default of 0600.

auto= This corresponds to the **--auto** flag to mdadm. Give **yes**, **md**, **mdp**, **part** — possibly followed by a number of partitions — to indicate how missing device entries should be created.

metadata=

The name of the metadata format to use if none is explicitly given. This can be useful to impose a system-wide default of version-1 superblocks.

symlinks=no

Normally when creating devices in **/dev/md/** *mdadm* will create a matching symlink from **/dev/** with a name starting **md** or **md_**. Give **symlinks=no** to suppress this symlink creation.

names=yes

Since Linux 2.6.29 it has been possible to create **md** devices with a name like **md_home** rather than just a number, like **md3**. *mdadm* will use the numeric alternative by default as other tools that interact with md arrays may expect only numbers. If **names=yes** is given in *mdadm.conf* then *mdadm* will use a name when appropriate. If **names=no** is given, then non-numeric *md* device names will not be used even if the default changes in a future release of *mdadm*.

bbl=no By default, *mdadm* will reserve space for a bad block list (bbl) on all devices included in or added to any array that supports them. Setting **bbl=no** will prevent this, so newly added devices will not have a bad block log.

HOMEHOST

The **homehost** line gives a default value for the **--homehost=** option to *mdadm*. There should normally be only one other word on the line. It should either be a host name, or one of the special words **<system>**, **<none>** and **<ignore>**. If **<system>** is given, then the **gethostname(2)** system-call is used to get the host name. This is the default.

If **<ignore>** is given, then a flag is set so that when arrays are being auto-assembled the checking of the recorded *homehost* is disabled. If **<ignore>** is given it is also possible to give an explicit name which will be used when creating arrays. This is the only case when there can be more than one other word on the **HOMEHOST** line. If there are other words, or other **HOMEHOST** lines, they are silently ignored.

If **<none>** is given, then the default of using **gethostname(2)** is over-ridden and no *homehost* name is assumed.

When arrays are created, this host name will be stored in the metadata. When arrays are assembled using auto-assembly, arrays which do not record the correct *homehost* name in their metadata will be assembled using a "foreign" name. A "foreign" name always ends with a digit string preceded by an underscore to differentiate it from any possible local name. e.g. **/dev/md/1_1** or **/dev/md/home_0**.

AUTO A list of names of metadata format can be given, each preceded by a plus or minus sign. Also the word *homehost* is allowed as is *all* preceded by plus or minus sign. *all* is usually last.

When *mdadm* is auto-assembling an array, either via *--assemble* or *--incremental* and it finds metadata of a given type, it checks that metadata type against those listed in this line. The first match wins, where *all* matches anything. If a match is found that was preceded by a plus sign, the auto assembly is allowed. If the match was preceded by a minus sign, the auto assembly is disallowed. If no match is found, the auto assembly is allowed.

If the metadata indicates that the array was created for *this* host, and the word *homehost* appears before any other match, then the array is treated as a valid candidate for auto-assembly.

This can be used to disable all auto-assembly (so that only arrays explicitly listed in mdadm.conf or on the command line are assembled), or to disable assembly of certain metadata types which might be handled by other software. It can also be used to disable assembly of all foreign arrays - normally such arrays are assembled but given a non-deterministic name in **/dev/md/**.

The known metadata types are **0.90**, **1.x**, **ddf**, **imsm**.

AUTO should be given at most once. Subsequent lines are silently ignored. Thus an earlier config file in a config directory will over-ride the setting in a later config file.

POLICY

This is used to specify what automatic behavior is allowed on devices newly appearing in the system and provides a way of marking spares that can be moved to other arrays as well as the migration domains. *Domain* can be defined through *policy* line by specifying a domain name for a number of paths from **/dev/disk/by-path/**. A device may belong to several domains. The domain of an array is a union of domains of all devices in that array. A spare can be automatically moved from one array to another if the set of the destination array's *domains* contains all the *domains* of the new disk or if both arrays have the same *spare-group*.

To update hot plug configuration it is necessary to execute **mdadm --udev-rules** command after changing the config file

Keywords used in the *POLICY* line and supported values are:

domain=

any arbitrary string

metadata=

0.9 1.x ddf or imsm

path= file glob matching anything from **/dev/disk/by-path**

type= either **disk** or **part**.

action=

include, re-add, spare, spare-same-slot, or force-spare

auto= yes, no, or homehost.

The *action* item determines the automatic behavior allowed for devices matching the *path* and *type* in the same line. If a device matches several lines with different *actions* then the most permissive will apply. The ordering of policy lines is irrelevant to the end result.

include

allows adding a disk to an array if metadata on that disk matches that array

re-add will include the device in the array if it appears to be a current member or a member that was recently removed and the array has a write-intent-bitmap to allow the **re-add** functionality.

spare as above and additionally: if the device is bare it can become a spare if there is any array that it is a candidate for based on domains and metadata.

spare=same-slot

as above and additionally if given slot was used by an array that went degraded recently and the device plugged in has no metadata then it will be automatically added to that array (or it's container)

force-spare

as above and the disk will become a spare in remaining cases

PART-POLICY

This is similar to **POLICY** and accepts the same keyword assignments. It allows a consistent set of policies to be applied to each of the partitions of a device.

A **PART-POLICY** line should set *type=disk* and identify the path to one or more disk devices. Each partition on these disks will be treated according to the *action=* setting from this line. If a *domain* is set in the line, then the domain associated with each partition will be based on the domain, but with "**-partN**" appended, when N is the partition number for the partition that was found.

SYSFS The **SYSFS** line lists custom values of MD device's sysfs attributes which will be stored in sysfs after the array is assembled. Multiple lines are allowed and each line has to contain the uid or the name of the device to which it relates.

uuid= hexadecimal identifier of MD device. This has to match the uuid stored in the superblock.

name= name of the MD device as was given to *mdadm* when the array was created. It will be ignored if **uuid** is not empty.

MONITORDELAY

The **monitordelay** line gives a delay in seconds *mdadm* shall wait before pooling md arrays when *mdadm* is running in **--monitor** mode. **-d**/**--delay** command line argument takes precedence over the config file

EXAMPLE

```
DEVICE /dev/sd[bcdjkl]1
DEVICE /dev/hda1 /dev/hdb1

# /dev/md0 is known by its UUID.
ARRAY /dev/md0 UUID=3aaa0122:29827cf:a5331ad66:ca767371
# /dev/md1 contains all devices with a minor number of
# 1 in the superblock.
ARRAY /dev/md1 superminor=1
# /dev/md2 is made from precisely these two devices
ARRAY /dev/md2 devices=/dev/hda1,/dev/hdb1

# /dev/md4 and /dev/md5 are a spare-group and spares
# can be moved between them
ARRAY /dev/md4 uuid=b23f3c6d:aec43a9f:fd65db85:369432df
    spare-group=group1
ARRAY /dev/md5 uuid=19464854:03f71b1b:e0df2edd:246cc977
    spare-group=group1
# /dev/md/home is created if need to be a partitionable md array
# any spare device number is allocated.
```

```
ARRAY /dev/md/home UUID=9187a482:5dde19d9:eea3cc4a:d646ab8b
    auto=part
# The name of this array contains a space.
ARRAY /dev/md9 name='Data Storage'

POLICY domain=domain1 metadata=imsm path=pci-0000:00:1f.2-scsi-*
    action=spare
POLICY domain=domain1 metadata=imsm path=pci-0000:04:00.0-scsi-[01]*
    action=include
# One domain comprising of devices attached to specified paths is defined.
# Bare device matching first path will be made an imsm spare on hot plug.
# If more than one array is created on devices belonging to domain1 and
# one of them becomes degraded, then any imsm spare matching any path for
# given domain name can be migrated.
MAILADDR root@mydomain.tld
PROGRAM /usr/sbin/handle-mdadm-events
CREATE group=system mode=0640 auto=part-8
HOMEHOST <system>
AUTO +1.x homehost -all
SYSFS name=/dev/md/raid5 group_thread_cnt=4 sync_speed_max=1000000
SYSFS uid=bead5eb6:31c17a27:da120ba2:7dfda40d group_thread_cnt=4 sync_speed_max=1000000
MONITORDELAY 60
```

SEE ALSO

mdadm(8), md(4).

NAME

mesg – display (or do not display) messages from other users

SYNOPSIS

mesg [option] [n|y]

DESCRIPTION

The **mesg** utility is invoked by a user to control write access others have to the terminal device associated with standard error output. If write access is allowed, then programs such as **talk(1)** and **write(1)** may display messages on the terminal.

Traditionally, write access is allowed by default. However, as users become more conscious of various security risks, there is a trend to remove write access by default, at least for the primary login shell. To make sure your ttys are set the way you want them to be set, **mesg** should be executed in your login scripts.

The **mesg** utility silently exits with error status 2 if not executed on terminal. In this case execute **mesg** is pointless. The command line option **--verbose** forces mesg to print a warning in this situation. This behaviour has been introduced in version 2.33.

ARGUMENTS

n

Disallow messages.

y

Allow messages to be displayed.

If no arguments are given, **mesg** shows the current message status on standard error output.

OPTIONS

-v, --verbose

Explain what is being done.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

EXIT STATUS

The **mesg** utility exits with one of the following values:

0

Messages are allowed.

1

Messages are not allowed.

>1

An error has occurred.

FILES

/dev/[pt]ty[pq]?

HISTORY

A **mesg** command appeared in Version 6 AT&T UNIX.

SEE ALSO

login(1), talk(1), write(1), wall(1), xterm(1)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **mesg** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

`mkdir` – make directories

SYNOPSIS

`mkdir [OPTION]... DIRECTORY...`

DESCRIPTION

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

`-m, --mode=MODE`

set file mode (as in `chmod`), not a=rwx – umask

`-p, --parents`

no error if existing, make parent directories as needed

`-v, --verbose`

print a message for each created directory

`-Z` set SELinux security context of each created directory to the default type

`--context[=CTX]`

like `-Z`, or if CTX is specified then set the SELinux or SMACK security context to CTX

`--help` display this help and exit

`--version`

output version information and exit

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`mkdir(2)`

Full documentation <<https://www.gnu.org/software/coreutils/mkdir>>
or available locally via: info '(coreutils) mkdir invocation'

NAME

mkdir, mkdirat – create a directory

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/stat.h>
int mkdir(const char *pathname, mode_t mode);
#include <fcntl.h>      /* Definition of AT_* constants */
#include <sys/stat.h>
int mkdirat(int dirfd, const char *pathname, mode_t mode);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
mkdirat():
Since glibc 2.10:
    _POSIX_C_SOURCE >= 200809L
Before glibc 2.10:
    _ATFILE_SOURCE
```

DESCRIPTION

mkdir() attempts to create a directory named *pathname*.

The argument *mode* specifies the mode for the new directory (see **inode(7)**). It is modified by the process's *umask* in the usual way: in the absence of a default ACL, the mode of the created directory is (*mode* & $\sim\text{umask}$ & 0777). Whether other *mode* bits are honored for the created directory depends on the operating system. For Linux, see NOTES below.

The newly created directory will be owned by the effective user ID of the process. If the directory containing the file has the set-group-ID bit set, or if the filesystem is mounted with BSD group semantics (*mount -o bsdgroups* or, synonymously *mount -o grpid*), the new directory will inherit the group ownership from its parent; otherwise it will be owned by the effective group ID of the process.

If the parent directory has the set-group-ID bit set, then so will the newly created directory.

mkdirat()

The **mkdirat()** system call operates in exactly the same way as **mkdir()**, except for the differences described here.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **mkdir()** for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **mkdir()**).

If *pathname* is absolute, then *dirfd* is ignored.

See **openat(2)** for an explanation of the need for **mkdirat()**.

RETURN VALUE

mkdir() and **mkdirat()** return zero on success. On error, -1 is returned and *errno* is set to indicate the error.

ERRORS

EACCES

The parent directory does not allow write permission to the process, or one of the directories in *pathname* did not allow search permission. (See also **path_resolution(7)**.)

EBADF

(**mkdirat()**) *pathname* is relative but *dirfd* is neither **AT_FDCWD** nor a valid file descriptor.

EDQUOT

The user's quota of disk blocks or inodes on the filesystem has been exhausted.

EEXIST

pathname already exists (not necessarily as a directory). This includes the case where *pathname* is a symbolic link, dangling or not.

EFAULT

pathname points outside your accessible address space.

EINVAL

The final component ("basename") of the new directory's *pathname* is invalid (e.g., it contains characters not permitted by the underlying filesystem).

ELOOP

Too many symbolic links were encountered in resolving *pathname*.

EMLINK

The number of links to the parent directory would exceed **LINK_MAX**.

ENAMETOOLONG

pathname was too long.

ENOENT

A directory component in *pathname* does not exist or is a dangling symbolic link.

ENOMEM

Insufficient kernel memory was available.

ENOSPC

The device containing *pathname* has no room for the new directory.

ENOSPC

The new directory cannot be created because the user's disk quota is exhausted.

ENOTDIR

A component used as a directory in *pathname* is not, in fact, a directory.

ENOTDIR

(**mkdirat()**) *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

EPERM

The filesystem containing *pathname* does not support the creation of directories.

EROFS

pathname refers to a file on a read-only filesystem.

VERSIONS

mkdirat() was added in Linux 2.6.16; library support was added in glibc 2.4.

STANDARDS

mkdir(): SVr4, BSD, POSIX.1-2001, POSIX.1-2008.

mkdirat(): POSIX.1-2008.

NOTES

Under Linux, apart from the permission bits, the **S_ISVTX mode** bit is also honored.

There are many infelicities in the protocol underlying NFS. Some of these affect **mkdir()**.

glibc notes

On older kernels where **mkdirat()** is unavailable, the glibc wrapper function falls back to the use of **mkdir()**. When *pathname* is a relative pathname, glibc constructs a pathname based on the symbolic link in */proc/self/fd* that corresponds to the *dirfd* argument.

SEE ALSO

mkdir(1), chmod(2), chown(2), mknod(2), mount(2), rmdir(2), stat(2), umask(2), unlink(2), acl(5), path_resolution(7)

NAME

mkfs – build a Linux filesystem

SYNOPSIS

mkfs [options] [**-t** *type*] [*fs-options*] *device* [*size*]

DESCRIPTION

This **mkfs** frontend is deprecated in favour of filesystem specific **mkfs.<type>** utils.

mkfs is used to build a Linux filesystem on a device, usually a hard disk partition. The *device* argument is either the device name (e.g., */dev/hda1*, */dev/sdb2*), or a regular file that shall contain the filesystem. The *size* argument is the number of blocks to be used for the filesystem.

The exit status returned by **mkfs** is 0 on success and 1 on failure.

In actuality, **mkfs** is simply a front-end for the various filesystem builders (**mkfs.fstype**) available under Linux. The filesystem-specific builder is searched for via your **PATH** environment setting only. Please see the filesystem-specific builder manual pages for further details.

OPTIONS

-t, --type *type*

Specify the *type* of filesystem to be built. If not specified, the default filesystem type (currently ext2) is used.

fs-options

Filesystem-specific options to be passed to the real filesystem builder.

-V, --verbose

Produce verbose output, including all filesystem-specific commands that are executed. Specifying this option more than once inhibits execution of any filesystem-specific commands. This is really only useful for testing.

-h, --help

Display help text and exit.

-V, --version

Print version and exit. (Option **-V** will display version information only when it is the only parameter, otherwise it will work as **--verbose**.)

BUGS

All generic options must precede and not be combined with filesystem-specific options. Some filesystem-specific programs do not automatically detect the device size and require the *size* parameter to be specified.

AUTHORS

David Engel <david@ods.com>, Fred N. van Kempen <waltje@uwalt.nl.mugnet.org>, Ron Sommeling <sommel@sci.kun.nl>.

The manual page was shamelessly adapted from Remy Card's version for the ext2 filesystem.

SEE ALSO

fs(5), **badblocks(8)**, **fsck(8)**, **mkdosfs(8)**, **mke2fs(8)**, **mkfs.bfs(8)**, **mkfs.ext2(8)**, **mkfs.ext3(8)**, **mkfs.ext4(8)**, **mkfs.minix(8)**, **mkfs.msdos(8)**, **mkfs.vfat(8)**, **mkfs.xfs(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **mkfs** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

mkfs.bfs – make an SCO bfs filesystem

SYNOPSIS

mkfs.bfs [options] *device* [*block-count*]

DESCRIPTION

mkfs.bfs creates an SCO bfs filesystem on a block device (usually a disk partition or a file accessed via the loop device).

The *block-count* parameter is the desired size of the filesystem, in blocks. If nothing is specified, the entire partition will be used.

OPTIONS

-N, --inodes *number*

Specify the desired *number* of inodes (at most 512). If nothing is specified, some default number in the range 48–512 is picked depending on the size of the partition.

-V, --vname *label*

Specify the volume *label*. I have no idea if/where this is used.

-F, --fname *name*

Specify the filesystem *name*. I have no idea if/where this is used.

-v, --verbose

Explain what is being done.

-c

This option is silently ignored.

-l

This option is silently ignored.

-h, --help

Display help text and exit.

-V, --version

Print version and exit. Option **-V** only works as **--version** when it is the only option.

EXIT STATUS

The exit status returned by **mkfs.bfs** is 0 when all went well, and 1 when something went wrong.

SEE ALSO

mkfs(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **mkfs.bfs** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

`mkfs.btrfs` – create a btrfs filesystem

SYNOPSIS

mkfs.btrfs [options] <*device*> [<*device*>...]

DESCRIPTION

mkfs.btrfs is used to create the btrfs filesystem on a single or multiple devices. <*device*> is typically a block device but can be a file-backed image as well. Multiple devices are grouped by UUID of the filesystem.

Before mounting such filesystem, the kernel module must know all the devices either via preceding execution of **btrfs device scan** or using the **device** mount option. See section **MULTIPLE DEVICES** for more details.

The default block group profiles for data and metadata depend on number of devices and possibly other factors. It's recommended to use specific profiles but the defaults should be OK and allowing future conversions to other profiles. Please see options **-d** and **-m** for further details and **btrfs-balance(8)** for the profile conversion post mkfs.

OPTIONS

-b|--byte-count <*size*>

Specify the size of the filesystem. If this option is not used, then `mkfs.btrfs` uses the entire device space for the filesystem.

--csum <*type*>, **--checksum** <*type*>

Specify the checksum algorithm. Default is *crc32c*. Valid values are *crc32c*, *xxhash*, *sha256* or *blake2*. To mount such filesystem kernel must support the checksums as well. See *CHECKSUM ALGORITHMS* in **btrfs(5)**.

-d|--data <*profile*>

Specify the profile for the data block groups. Valid values are *raid0*, *raid1*, *raid1c3*, *raid1c4*, *raid5*, *raid6*, *raid10* or *single* or *dup* (case does not matter).

See *DUP PROFILES ON A SINGLE DEVICE* for more details.

On multiple devices, the default was *raid0* until version 5.7, while it is *single* since version 5.8. You can still select *raid0* manually, but it was not suitable as default.

-m|--metadata <*profile*>

Specify the profile for the metadata block groups. Valid values are *raid0*, *raid1*, *raid1c3*, *raid1c4*, *raid5*, *raid6*, *raid10*, *single* or *dup* (case does not matter).

Default on a single device filesystem is *DUP* and is recommended for metadata in general. The duplication might not be necessary in some use cases and it's up to the user to change that at mkfs time or later. This depends on hardware that could potentially deduplicate the blocks again but this cannot be detected at mkfs time.

NOTE

Up to version 5.14 there was a detection of a SSD device (more precisely if it's a rotational device, determined by the contents of file `/sys/block/DEV/queue/rotational`) that used to select *single*. This has changed in version 5.15 to be always *dup*.

Note that the rotational status can be arbitrarily set by the underlying block device driver and may not reflect the true status (network block device, memory-backed SCSI devices, real block device behind some additional device mapper layer, etc). It's recommended to always set the options **--data**/**--metadata** to avoid confusion and unexpected results.

See *DUP PROFILES ON A SINGLE DEVICE* for more details.

On multiple devices the default is *raid1*.

-M|--mixed

Normally the data and metadata block groups are isolated. The *mixed* mode will remove the isolation and store both types in the same block group type. This helps to utilize the free space regardless of the purpose and is suitable for small devices. The separate allocation of block groups leads to a situation where the space is reserved for the other block group type, is not available for allocation and can lead to ENOSPC state.

The recommended size for the mixed mode is for filesystems less than 1GiB. The soft recommendation is to use it for filesystems smaller than 5GiB. The mixed mode may lead to degraded performance on larger filesystems, but is otherwise usable, even on multiple devices.

The *nodesize* and *sectorsize* must be equal, and the block group types must match.

Note

versions up to 4.2.x forced the mixed mode for devices smaller than 1GiB. This has been removed in 4.3+ as it caused some usability issues.

-l|--leafsize <size>

Alias for *--nodesize*. Deprecated.

-n|--nodesize <size>

Specify the *nodesize*, the tree block size in which btrfs stores metadata. The default value is 16KiB (16384) or the page size, whichever is bigger. Must be a multiple of the *sectorsize* and a power of 2, but not larger than 64KiB (65536). Leafsize always equals *nodesize* and the options are aliases.

Smaller node size increases fragmentation but leads to taller b-trees which in turn leads to lower locking contention. Higher node sizes give better packing and less fragmentation at the cost of more expensive memory operations while updating the metadata blocks.

Note

versions up to 3.11 set the *nodesize* to 4k.

-s|--sectorsize <size>

Specify the *sectorsize*, the minimum data block allocation unit.

The default value is the page size and is autodetected. If the *sectorsize* differs from the page size, the created filesystem may not be mountable by the running kernel. Therefore it is not recommended to use this option unless you are going to mount it on a system with the appropriate page size.

-L|--label <string>

Specify a label for the filesystem. The *string* should be less than 256 bytes and must not contain newline characters.

-K|--nodiscard

Do not perform whole device TRIM operation on devices that are capable of that. This does not affect discard/trim operation when the filesystem is mounted. Please see the mount option *discard* for that in **btrfs(5)**.

-r|--rootdir <rootdir>

Populate the toplevel subvolume with files from *rootdir*. This does not require root permissions to write the new files or to mount the filesystem.

Note

This option may enlarge the image or file to ensure it's big enough to contain the files from *rootdir*. Since version 4.14.1 the filesystem size is not minimized. Please see option *--shrink* if you need that functionality.

--shrink

Shrink the filesystem to its minimal size, only works with *--rootdir* option.

If the destination block device is a regular file, this option will also truncate the file to the minimal size. Otherwise it will reduce the filesystem available space. Extra space will not be usable unless the filesystem is mounted and resized using *btrfs filesystem resize*.

Note

prior to version 4.14.1, the shrinking was done automatically.

-O|—features <feature1>[,<feature2>...]

A list of filesystem features turned on at mkfs time. Not all features are supported by old kernels. To disable a feature, prefix it with ^.

See section **FILESYSTEM FEATURES** for more details. To see all available features that mkfs.btrfs supports run:

mkfs.btrfs -O list-all

-R|—runtime-features <feature1>[,<feature2>...]

A list of features that can be enabled at mkfs time, otherwise would have to be turned on a mounted filesystem. Although no runtime feature is enabled by default, to disable a feature, prefix it with ^.

See section **RUNTIME FEATURES** for more details. To see all available runtime features that mkfs.btrfs supports run:

mkfs.btrfs -R list-all

-f|—force

Forcibly overwrite the block devices when an existing filesystem is detected. By default, mkfs.btrfs will utilize *libblkid* to check for any known filesystem on the devices. Alternatively you can use the **wipefs** utility to clear the devices.

-q|—quiet

Print only error or warning messages. Options --features or --help are unaffected. Resets any previous effects of --verbose.

-U|—uuid <UUID>

Create the filesystem with the given *UUID*. The UUID must not exist on any filesystem currently present.

-v|—verbose

Increase verbosity level, default is 1.

-V|—version

Print the **mkfs.btrfs** version and exit.

--help

Print help.

SIZE UNITS

The default unit is *byte*. All size parameters accept suffixes in the 1024 base. The recognized suffixes are: *k*, *m*, *g*, *t*, *p*, *e*, both uppercase and lowercase.

MULTIPLE DEVICES

Before mounting a multiple device filesystem, the kernel module must know the association of the block devices that are attached to the filesystem **UUID**.

There is typically no action needed from the user. On a system that utilizes a udev-like daemon, any new block device is automatically registered. The rules call **btrfs device scan**.

The same command can be used to trigger the device scanning if the btrfs kernel module is reloaded (naturally all previous information about the device registration is lost).

Another possibility is to use the mount options **device** to specify the list of devices to scan at the time of mount.

```
# mount -o device=/dev/sdb,device=/dev/sdc /dev/sda /mnt
```

Note

that this means only scanning, if the devices do not exist in the system, mount will fail anyway. This can happen on systems without initramfs/initrd and root partition created with RAID1/10/5/6 profiles. The mount action can happen before all block devices are discovered. The waiting is usually done on the initramfs/initrd systems.

RAID5/6 has known problems and should not be used in production.

FILESYSTEM FEATURES

Features that can be enabled during creation time. See also **btrfs(5)** section *FILESYSTEM FEATURES*.

mixed-bg

(kernel support since 2.6.37)

mixed data and metadata block groups, also set by option *--mixed*

extref

(default since btrfs-progs 3.12, kernel support since 3.7)

increased hardlink limit per file in a directory to 65536, older kernels supported a varying number of hardlinks depending on the sum of all file name sizes that can be stored into one metadata block

raid56

(kernel support since 3.9)

extended format for RAID5/6, also enabled if raid5 or raid6 block groups are selected

skinny-metadata

(default since btrfs-progs 3.18, kernel support since 3.10)

reduced-size metadata for extent references, saves a few percent of metadata

no-holes

(default since btrfs-progs 5.15, kernel support since 3.14)

improved representation of file extents where holes are not explicitly stored as an extent, saves a few percent of metadata if sparse files are used

zoned

(kernel support since 5.12)

zoned mode, data allocation and write friendly to zoned/SMR/ZBC/ZNS devices, see *ZONED MODE* in **btrfs(5)**, the mode is automatically selected when a zoned device is detected

RUNTIME FEATURES

Features that are typically enabled on a mounted filesystem, eg. by a mount option or by an ioctl. Some of them can be enabled early, at mkfs time. This applies to features that need to be enabled once and then the status is permanent, this does not replace mount options.

quota

(kernel support since 3.4)

Enable quota support (qgroups). The qgroup accounting will be consistent, can be used together with *--rootdir*. See also **btrfs-quota(8)**.

free-space-tree

(default since btrfs-progs 5.15, kernel support since 4.5)

Enable the free space tree (mount option space_cache=v2) for persisting the free space cache.

BLOCK GROUPS, CHUNKS, RAID

The highlevel organizational units of a filesystem are block groups of three types: data, metadata and system.

DATA

store data blocks and nothing else

METADATA

store internal metadata in b-trees, can store file data if they fit into the inline limit

SYSTEM

store structures that describe the mapping between the physical devices and the linear logical space representing the filesystem

Other terms commonly used:

block group, chunk

a logical range of space of a given profile, stores data, metadata or both; sometimes the terms are used interchangeably

A typical size of metadata block group is 256MiB (filesystem smaller than 50GiB) and 1GiB (larger than 50GiB), for data it's 1GiB. The system block group size is a few megabytes.

RAID

a block group profile type that utilizes RAID-like features on multiple devices: striping, mirroring, parity

profile

when used in connection with block groups refers to the allocation strategy and constraints, see the section *PROFILES* for more details

PROFILES

There are the following block group types available:

Profile	Redundancy			Space utilization	Min/max devices
	Copies	Parity	Striping		
single	1			100%	1/any
DUP	2 / 1 device			50%	1/any ^(see note 1)
RAID0			1 to N	100%	1/any ^(see note 5)
RAID1	2			50%	2/any
RAID1C3	3			33%	3/any
RAID1C4	4			25%	4/any
RAID10	2		1 to N	50%	2/any ^(see note 5)
RAID5	1	1	2 to N-1	(N-1)/N	2/any ^(see note 2)
RAID6	1	2	3 to N-2	(N-2)/N	3/any ^(see note 3)

Warning

It's not recommended to create filesystems with RAID0/1/10/5/6 profiles on partitions from the same device. Neither redundancy nor performance will be improved.

Note 1: DUP may exist on more than 1 device if it starts on a single device and another one is added. Since version 4.5.1, **mkfs.btrfs** will let you create DUP on multiple devices without restrictions.

Note 2: It's not recommended to use 2 devices with RAID5. In that case, parity stripe will contain the same data as the data stripe, making RAID5 degraded to RAID1 with more overhead.

Note 3: It's also not recommended to use 3 devices with RAID6, unless you want to get effectively 3 copies in a RAID1-like manner (but not exactly that).

Note 4: Since kernel 5.5 it's possible to use RAID1C3 as replacement for RAID6, higher space cost but reliable.

Note 5: Since kernel 5.15 it's possible to use (mount, convert profiles) RAID0 on one device and RAID10 on two devices.

PROFILE LAYOUT

For the following examples, assume devices numbered by 1, 2, 3 and 4, data or metadata blocks A, B, C, D, with possible stripes eg. A1, A2 that would be logically A, etc. For parity profiles PA and QA are parity and syndrom, associated with the given stripe. The simple layouts single or DUP are left out. Actual physical block placement on devices depends on current state of the free/allocated space and may appear random. All devices are assumed to be present at the time of the blocks would have been written.

RAID1

device 1	device 2	device 3	device 4
A	D		
B			C
C			
D	A	B	

RAID1C3

device 1	device 2	device 3	device 4
A	A	D	
B		B	
C		A	C
D	D	C	B

RAID0

device 1	device 2	device 3	device 4
A2	C3	A3	C2
B1	A1	D2	B3
C1	D3	B4	D1
D4	B2	C4	A4

RAID5

device 1	device 2	device 3	device 4
A2	C3	A3	C2
B1	A1	D2	B3
C1	D3	PB	D1
PD	B2	PC	PA

RAID6

device 1	device 2	device 3	device 4
A2	QC	QA	C2
B1	A1	D2	QB
C1	QD	PB	D1
PD	B2	PC	PA

DUP PROFILES ON A SINGLE DEVICE

The mkfs utility will let the user create a filesystem with profiles that write the logical blocks to 2 physical locations. Whether there are really 2 physical copies highly depends on the underlying device type.

For example, a SSD drive can remap the blocks internally to a single copy—thus deduplicating them. This negates the purpose of increased redundancy and just wastes filesystem space without providing the expected level of redundancy.

The duplicated data/metadata may still be useful to statistically improve the chances on a device that might perform some internal optimizations. The actual details are not usually disclosed by vendors. For example we could expect that not all blocks get deduplicated. This will provide a non-zero probability of recovery compared to a zero chance if the single profile is used. The user should make the tradeoff decision. The deduplication in SSDs is thought to be widely available so the reason behind the mkfs default is to not give a false sense of redundancy.

As another example, the widely used USB flash or SD cards use a translation layer between the logical and physical view of the device. The data lifetime may be affected by frequent plugging. The memory cells could get damaged, hopefully not destroying both copies of particular data in case of DUP.

The wear levelling techniques can also lead to reduced redundancy, even if the device does not do any deduplication. The controllers may put data written in a short timespan into the same physical storage unit (cell, block etc). In case this unit dies, both copies are lost. BTRFS does not add any artificial delay between metadata writes.

The traditional rotational hard drives usually fail at the sector level.

In any case, a device that starts to misbehave and repairs from the DUP copy should be replaced! **DUP is not backup.**

KNOWN ISSUES

SMALL FILESYSTEMS AND LARGE NODESIZE

The combination of small filesystem size and large nodesize is not recommended in general and can lead to various ENOSPC-related issues during mount time or runtime.

Since mixed block group creation is optional, we allow small filesystem instances with differing values for *sectorsize* and *nodesize* to be created and could end up in the following situation:

```
# mkfs.btrfs -f -n 65536 /dev/loop0
btrfs-progs v3.19-rc2-405-g976307c
See http://btrfs.wiki.kernel.org for more information.
```

Performing full device TRIM (512.00MiB) ...
Label: (null)

```
UUID:      49fab72e-0c8b-466b-a3ca-d1bfe56475f0
Node size: 65536
Sector size: 4096
Filesystem size: 512.00MiB
Block group profiles:
Data:      single     8.00MiB
Metadata: DUP        40.00MiB
System:    DUP        12.00MiB
SSD detected: no
Incompat features: extref, skinny-metadata
Number of devices: 1
Devices:
ID      SIZE PATH
1  512.00MiB /dev/loop0

# mount /dev/loop0 /mnt/
mount: mount /dev/loop0 on /mnt failed: No space left on device
```

The ENOSPC occurs during the creation of the UUID tree. This is caused by large metadata blocks and space reservation strategy that allocates more than can fit into the filesystem.

AVAILABILITY

mkfs.btrfs is part of btrfs-progs. Please refer to the btrfs wiki <http://btrfs.wiki.kernel.org> for further details.

SEE ALSO

btrfs(5), **btrfs(8)**, **btrfs–balance(8)**, **wipefs(8)**

NAME

mkfs.exfat – create an exFAT filesystem

SYNOPSIS

```
mkfs.exfat [ -b boundary_alignment ] [ -c cluster_size ] [ -f ] [ -h ] [ -L volume_label ] [ --pack-bitmap ] [ -v ] device
mkfs.exfat -V
```

DESCRIPTION

mkfs.exfat creates an exFAT filesystem by writing on a special file using the values found in the arguments of the command line. It is invoked automatically by **mkfs(8)** when it is given the **-t exfat** option.

As an example, to make a filesystem on the first partition on the first SCSI disk, use:

```
mkfs.exfat /dev/sda1
```

OPTIONS

-b, --boundary-align=alignment

Specifies the alignment for the FAT and the start of the cluster heap. The *alignment* argument is specified in bytes or may be specified with **m/M** suffix for mebibytes or **k/K** suffix for kibibytes and should be a power of two. Some media like SD cards need this for optimal performance and endurance, in which case *alignment* should be set to half of the card's native boundary unit size. If the card's native boundary unit size is not known, refer to the following table of boundary unit sizes recommended by the SD Card Association.

Card Capacity Range	Cluster Size	Boundary Unit
$\leq 8 \text{ MiB}$	8 KiB	8 KiB
$>8 \text{ MiB} \leq 64 \text{ MiB}$	16 KiB	16 KiB
$>64 \text{ MiB} \leq 256 \text{ MiB}$	16 KiB	32 KiB
$>256 \text{ MiB} \leq 1 \text{ GiB}$	16 KiB	64 KiB
$>1 \text{ GiB} \leq 2 \text{ GiB}$	32 KiB	64 KiB
$>2 \text{ GiB} \leq 32 \text{ GiB}$	32 KiB	4 MiB
$>32 \text{ GiB} \leq 128 \text{ GiB}$	128 KiB	16 MiB
$>128 \text{ GiB} \leq 512 \text{ GiB}$	256 KiB	32 MiB
$>512 \text{ GiB} \leq 2 \text{ TiB}$	512 KiB	64 MiB

-c, --cluster-size=size

Specifies the cluster size of the exFAT file system. The *size* argument is specified in bytes or may be specified with **m/M** suffix for mebibytes or **k/K** suffix for kibibytes and must be a power of two.

-f, --full-format

Performs a full format. This zeros the entire disk device while creating the exFAT filesystem.

-h, --help

Prints the help and exit.

-L, --volume-label=label

Specifies the volume label to be associated with the exFAT filesystem.

--pack-bitmap

Attempts to relocate the exFAT allocation bitmap so that it ends at the alignment boundary immediately following the FAT rather than beginning at that boundary. This strictly violates the SD card specification but may improve performance and endurance on SD cards and other flash media not designed for use with exFAT by allowing file-system metadata updates to touch fewer flash allocation units. Furthermore, many SD cards and other flash devices specially optimize the allocation unit where the FAT resides so as to support tiny writes with reduced write amplification but expect only larger writes in subsequent allocation units — where the exFAT bitmap would be placed by default. Specifying **--pack-bitmap** attempts to avoid the potential problems associated with issuing many small writes to the bitmap by making it share an allocation unit with the FAT. If there is insufficient space for the bitmap there, then this option will have no effect, and the

bitmap will be aligned at the boundary as by default.

-v, --verbose

Prints verbose debugging information while creating the exFAT filesystem.

-V, --version

Prints the version number and exits.

SEE ALSO

mkfs(8), mount(8),

NAME

mke2fs – create an ext2/ext3/ext4 file system

SYNOPSIS

```
mke2fs [ -c | -I filename ] [ -b block-size ] [ -C cluster-size ] [ -d root-directory ] [ -D ] [ -g blocks-per-group ] [ -G number-of-groups ] [ -i bytes-per-inode ] [ -I inode-size ] [ -j ] [ -J journal-options ] [ -N number-of-inodes ] [ -n ] [ -m reserved-blocks-percentage ] [ -o creator-os ] [ -O [^]feature[,...] ] [ -q ] [ -r fs-revision-level ] [ -E extended-options ] [ -v ] [ -F ] [ -L volume-label ] [ -M last-mounted-directory ] [ -S ] [ -t fs-type ] [ -T usage-type ] [ -U UUID ] [ -V ] [ -e errors-behavior ] [ -z undo_file ] device [ fs-size ]
```

```
mke2fs -O journal_dev [ -b block-size ] [ -L volume-label ] [ -n ] [ -q ] [ -v ] external-journal [ fs-size ]
```

DESCRIPTION

mke2fs is used to create an ext2, ext3, or ext4 file system, usually in a disk partition (or file) named by *device*.

The file system size is specified by *fs-size*. If *fs-size* does not have a suffix, it is interpreted as power-of-two kilobytes, unless the **-b** *blocksize* option is specified, in which case *fs-size* is interpreted as the number of *blocksize* blocks. If the *fs-size* is suffixed by 'k', 'm', 'g', 't' (either upper-case or lower-case), then it is interpreted in power-of-two kilobytes, megabytes, gigabytes, terabytes, etc. If *fs-size* is omitted, **mke2fs** will create the file system based on the device size.

If **mke2fs** is run as **mkfs.XXX** (i.e., **mkfs.ext2**, **mkfs.ext3**, or **mkfs.ext4**) the option **-t XXX** is implied; so **mkfs.ext3** will create a file system for use with ext3, **mkfs.ext4** will create a file system for use with ext4, and so on.

The defaults of the parameters for the newly created file system, if not overridden by the options listed below, are controlled by the */etc/mke2fs.conf* configuration file. See the **mke2fs.conf(5)** manual page for more details.

OPTIONS**-b** *block-size*

Specify the size of blocks in bytes. Valid block-size values are powers of two from 1024 up to 65536 (however note that the kernel is able to mount only file systems with block-size smaller or equal to the system page size - 4k on x86 systems, up to 64k on ppc64 or aarch64 depending on kernel configuration). If omitted, block-size is heuristically determined by the file system size and the expected usage of the file system (see the **-T** option). In most common cases, the default block size is 4k. If *block-size* is preceded by a negative sign ('-'), then **mke2fs** will use heuristics to determine the appropriate block size, with the constraint that the block size will be at least *block-size* bytes. This is useful for certain hardware devices which require that the *blocksize* be a multiple of 2k.

-c Check the device for bad blocks before creating the file system. If this option is specified twice, then a slower read-write test is used instead of a fast read-only test.

-C *cluster-size*

Specify the size of cluster in bytes for file systems using the bigalloc feature. Valid cluster-size values are from 2048 to 256M bytes per cluster. This can only be specified if the bigalloc feature is enabled. (See the **ext4 (5)** man page for more details about bigalloc.) The default cluster size if bigalloc is enabled is 16 times the block size.

-d *root-directory*

Copy the contents of the given directory into the root directory of the file system.

-D Use direct I/O when writing to the disk. This avoids mke2fs dirtying a lot of buffer cache memory, which may impact other applications running on a busy server. This option will cause mke2fs to run much more slowly, however, so there is a tradeoff to using direct I/O.

-e error-behavior

Change the behavior of the kernel code when errors are detected. In all cases, a file system error will cause **e2fsck(8)** to check the file system on the next boot. *error-behavior* can be one of the following:

continue	Continue normal execution.
remount-ro	Remount file system read-only.
panic	Cause a kernel panic.

-E extended-options

Set extended options for the file system. Extended options are comma separated, and may take an argument using the equals ('=') sign. The **-E** option used to be **-R** in earlier versions of **mke2fs**. The **-R** option is still accepted for backwards compatibility, but is deprecated. The following extended options are supported:

encoding=encoding-name

Enable the *casemap* feature in the super block and set *encoding-name* as the encoding to be used. If *encoding-name* is not specified, the encoding defined in **mke2fs.conf(5)** is used.

encoding_flags=encoding-flags

Define parameters for file name character encoding operations. If a flag is not changed using this parameter, its default value is used. *encoding-flags* should be a comma-separated lists of flags to be enabled. To disable a flag, add it to the list with the prefix "no".

The only flag that can be set right now is *strict* which means that invalid strings should be rejected by the file system. In the default configuration, the *strict* flag is disabled.

mmp_update_interval=interval

Adjust the initial MMP update interval to *interval* seconds. Specifying an *interval* of 0 means to use the default interval. The specified interval must be less than 300 seconds. Requires that the **mmp** feature be enabled.

stride=stride-size

Configure the file system for a RAID array with *stride-size* file system blocks. This is the number of blocks read or written to disk before moving to the next disk, which is sometimes referred to as the *chunk size*. This mostly affects placement of file system metadata like bitmaps at **mke2fs** time to avoid placing them on a single disk, which can hurt performance. It may also be used by the block allocator.

stripe_width=stripe-width

Configure the file system for a RAID array with *stripe-width* file system blocks per stripe. This is typically *stride-size* * N, where N is the number of data-bearing disks in the RAID (e.g. for RAID 5 there is one parity disk, so N will be the number of disks in the array minus 1). This allows the block allocator to prevent read-modify-write of the parity in a RAID stripe if possible when the data is written.

offset=offset

Create the file system at an offset from the beginning of the device or file. This can be useful when creating disk images for virtual machines.

resize=max-online-resize

Reserve enough space so that the block group descriptor table can grow to support a file system that has *max-online-resize* blocks.

lazy_itable_init[= <0 to disable, 1 to enable>]

If enabled and the uninit_bg feature is enabled, the inode table will not be fully initialized by **mke2fs**. This speeds up file system initialization noticeably, but it requires the kernel to finish initializing the file system in the background when the file system is first mounted. If the option value is omitted, it defaults to 1 to enable lazy inode table zeroing.

lazy_journal_init[= <0 to disable, 1 to enable>]

If enabled, the journal inode will not be fully zeroed out by **mke2fs**. This speeds up file system initialization noticeably, but carries some small risk if the system crashes before the journal has been overwritten entirely one time. If the option value is omitted, it defaults to 1 to enable lazy journal inode zeroing.

no_copy_xattrs

Normally **mke2fs** will copy the extended attributes of the files in the directory hierarchy specified via the (optional) **-d** option. This will disable the copy and leaves the files in the newly created file system without any extended attributes.

num_backup_sb=<0/1/2>

If the **sparse_super2** file system feature is enabled this option controls whether there will be 0, 1, or 2 backup superblocks created in the file system.

packed_meta_blocks[= <0 to disable, 1 to enable>]

Place the allocation bitmaps and the inode table at the beginning of the disk. This option requires that the flex_bg file system feature to be enabled in order for it to have effect, and will also create the journal at the beginning of the file system. This option is useful for flash devices that use SLC flash at the beginning of the disk. It also maximizes the range of contiguous data blocks, which can be useful for certain specialized use cases, such as supported Shingled Drives.

root_owner[=uid:gid]

Specify the numeric user and group ID of the root directory. If no UID:GID is specified, use the user and group ID of the user running **mke2fs**. In **mke2fs** 1.42 and earlier the UID and GID of the root directory were set by default to the UID and GID of the user running the mke2fs command. The **root_owner=** option allows explicitly specifying these values, and avoid side-effects for users that do not expect the contents of the file system to change based on the user running **mke2fs**.

test_fs Set a flag in the file system superblock indicating that it may be mounted using experimental kernel code, such as the ext4dev file system.

discard

Attempt to discard blocks at mkfs time (discarding blocks initially is useful on solid state devices and sparse / thin-provisioned storage). When the device advertises that discard also zeroes data (any subsequent read after the discard and before write returns zero), then mark all not-yet-zeroed inode tables as zeroed. This significantly speeds up file system initialization. This is set as default.

nodiscard

Do not attempt to discard blocks at mkfs time.

quotatype

Specify the which quota types (usrquota, grpquota, prjquota) which should be enabled in the created file system. The argument of this extended option should be a colon separated list. This option has effect only if the **quota**

feature is set. The default quota types to be initialized if this option is not specified is both user and group quotas. If the project feature is enabled that project quotas will be initialized as well.

- F** Force **mke2fs** to create a file system, even if the specified device is not a partition on a block special device, or if other parameters do not make sense. In order to force **mke2fs** to create a file system even if the file system appears to be in use or is mounted (a truly dangerous thing to do), this option must be specified twice.

-g *blocks-per-group*

Specify the number of blocks in a block group. There is generally no reason for the user to ever set this parameter, as the default is optimal for the file system. (For administrators who are creating file systems on RAID arrays, it is preferable to use the *stride* RAID parameter as part of the **-E** option rather than manipulating the number of blocks per group.) This option is generally used by developers who are developing test cases.

If the bigalloc feature is enabled, the **-g** option will specify the number of clusters in a block group.

-G *number-of-groups*

Specify the number of block groups that will be packed together to create a larger virtual block group (or "flex_bg group") in an ext4 file system. This improves meta-data locality and performance on meta-data heavy workloads. The number of groups must be a power of 2 and may only be specified if the **flex_bg** file system feature is enabled.

-i *bytes-per-inode*

Specify the bytes/inode ratio. **mk e2fs** creates an inode for every *bytes-per-inode* bytes of space on the disk. The larger the *bytes-per-inode* ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the file system, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to change this ratio on a file system after it is created, so be careful deciding the correct value for this parameter. Note that resizing a file system changes the number of inodes to maintain this ratio.

-I *inode-size*

Specify the size of each inode in bytes. The *inode-size* value must be a power of 2 larger or equal to 128. The larger the *inode-size* the more space the inode table will consume, and this reduces the usable space in the file system and can also negatively impact performance. It is not possible to change this value after the file system is created.

File systems with an inode size of 128 bytes do not support timestamps beyond January 19, 2038. Inodes which are 256 bytes or larger will support extended timestamps, project id's, and the ability to store some extended attributes in the inode table for improved performance.

The default inode size is controlled by the **mke2fs.conf(5)** file. In the **mke2fs.conf** file shipped with e2fsprogs, the default inode size is 256 bytes for most file systems, except for small file systems where the inode size will be 128 bytes.

- j** Create the file system with an ext3 journal. If the **-J** option is not specified, the default journal parameters will be used to create an appropriately sized journal (given the size of the file system) stored within the file system. Note that you must be using a kernel which has ext3 support in order to actually make use of the journal.

-J *journal-options*

Create the ext3 journal using options specified on the command-line. Journal options are comma separated, and may take an argument using the equals ('=') sign. The following journal options are supported:

size=*journal-size*

Create an internal journal (i.e., stored inside the file system) of size *journal-size* megabytes. The size of the journal must be at least 1024 file system blocks (i.e., 1MB if using 1k blocks, 4MB if using 4k blocks, etc.) and may

be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller)

fast_commit_size=*fast-commit-size*

Create an additional fast commit journal area of size *fast-commit-size* kilobytes. This option is only valid if **fast_commit** feature is enabled on the file system. If this option is not specified and if **fast_commit** feature is turned on, fast commit area size defaults to *journal-size* / 64 me gabytes. The total size of the journal with **fast_commit** feature set is *journal-size* + (*fast-commit-size* * 1024) megabytes. The total journal size may be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller).

location=*journal-location*

Specify the location of the journal. The argument *journal-location* can either be specified as a block number, or if the number has a units suffix (e.g., 'M', 'G', etc.) interpret it as the offset from the beginning of the file system.

device=*external-journal*

Attach the file system to the journal block device located on *external-journal*. The external journal must already have been created using the command

mke2fs -O journal_dev *external-journal*

Note that *external-journal* must have been created with the same block size as the new file system. In addition, while there is support for attaching multiple file systems to a single external journal, the Linux kernel and **e2fsck(8)** do not currently support shared external journals yet.

Instead of specifying a device name directly, *external-journal* can also be specified by either **LABEL=***label* or **UUID=***UUID* to locate the external journal by either the volume label or UUID stored in the ext2 superblock at the start of the journal. Use **dumpe2fs(8)** to display a journal device's volume label and UUID. See also the **-L** option of **tune2fs(8)**.

Only one of the **size** or **device** options can be given for a file system.

-l *filename*

Read the bad blocks list from *filename*. Note that the block numbers in the bad block list must be generated using the same block size as used by **mke2fs**. As a result, the **-c** option to **mk e2fs** is a much simpler and less error-prone method of checking a disk for bad blocks before formatting it, as **mke2fs** will automatically pass the correct parameters to the **badblocks** program.

-L *new-volume-label*

Set the volume label for the file system to *new-volume-label*. The maximum length of the volume label is 16 bytes.

-m *reserved-blocks-percentage*

Specify the percentage of the file system blocks reserved for the super-user. This avoids fragmentation, and allows root-owned daemons, such as **syslogd(8)**, to continue to function correctly after non-privileged processes are prevented from writing to the file system. The default percentage is 5%.

-M *last-mounted-directory*

Set the last mounted directory for the file system. This might be useful for the sake of utilities that key off of the last mounted directory to determine where the file system should be mounted.

- n** Causes **mke2fs** to not actually create a file system, but display what it would do if it were to create a file system. This can be used to determine the location of the backup superblocks for a particular file system, so long as the **mke2fs** parameters that were passed when the file system was originally created are used again. (With the **-n** option added, of course!)

-N number-of-inodes

Overrides the default calculation of the number of inodes that should be reserved for the file system (which is based on the number of blocks and the *bytes-per-inode* ratio). This allows the user to specify the number of desired inodes directly.

-o creator-os

Overrides the default value of the "creator operating system" field of the file system. The creator field is set by default to the name of the OS the **mke2fs** executable was compiled for.

-O []feature[,...]

Create a file system with the given features (file system options), overriding the default file system options. The features that are enabled by default are specified by the *base_features* relation, either in the *[defaults]* section in the **/etc/mke2fs.conf** configuration file, or in the *[fs_types]* subsections for the usage types as specified by the **-T** option, further modified by the *features* relation found in the *[fs_types]* subsections for the file system and usage types. See the **mke2fs.conf(5)** manual page for more details. The file system type-specific configuration setting found in the *[fs_types]* section will override the global default found in *[defaults]*.

The file system feature set will be further edited using either the feature set specified by this option, or if this option is not given, by the *default_features* relation for the file system type being created, or in the *[defaults]* section of the configuration file.

The file system feature set is comprised of a list of features, separated by commas, that are to be enabled. To disable a feature, simply prefix the feature name with a caret (^) character. Features with dependencies will not be removed successfully. The pseudo-file system feature "none" will clear all file system features.

For more information about the features which can be set, please see the manual page **ext4(5)**.

-q Quiet execution. Useful if **mk e2fs** is run in a script.**-r revision**

Set the file system revision for the new file system. Note that 1.2 kernels only support revision 0 file systems. The default is to create revision 1 file systems.

-S

Write superblock and group descriptors only. This is an extreme measure to be taken only in the very unlikely case that all of the superblock and backup superblocks are corrupted, and a last-ditch recovery method is desired by experienced users. It causes **mke2fs** to reinitialize the superblock and group descriptors, while not touching the inode table and the block and inode bitmaps. The **e2fsck** program should be run immediately after this option is used, and there is no guarantee that any data will be salvageable. Due to the wide variety of possible options to **mke2fs** that affect the on-disk layout, it is critical to specify exactly the same format options, such as blocksize, fs-type, feature flags, and other tunables when using this option, or the file system will be further corrupted. In some cases, such as file systems that have been resized, or have had features enabled after format time, it is impossible to overwrite all of the superblocks correctly, and at least some file system corruption will occur. It is best to run this on a full copy of the file system so other options can be tried if this doesn't work.

-t fs-type

Specify the file system type (i.e., ext2, ext3, ext4, etc.) that is to be created. If this option is not specified, **mke2fs** will pick a default either via how the command was run (for example, using a name of the form **mkfs.ext2**, **mkfs.ext3**, etc.) or via a default as defined by the **/etc/mke2fs.conf** file. This option controls which file system options are used by default, based on the **fstypes** configuration stanza in **/etc/mke2fs.conf**.

If the **-O** option is used to explicitly add or remove file system options that should be set in the newly created file system, the resulting file system may not be supported by the requested *fs-type*. (e.g., "**mke2fs -t ext3 -O extent /dev/sdXX**" will create a file system that is not supported by the

ext3 implementation as found in the Linux kernel; and "**mke2fs -t ext3 -O ^has_journal /dev/hdXX**" will create a file system that does not have a journal and hence will not be supported by the ext3 file system code in the Linux kernel.)

-T usage-type[,...]

Specify how the file system is going to be used, so that **mke2fs** can choose optimal file system parameters for that use. The usage types that are supported are defined in the configuration file **/etc/mke2fs.conf**. The user may specify one or more usage types using a comma separated list.

If this option is not specified, **mke2fs** will pick a single default usage type based on the size of the file system to be created. If the file system size is less than 3 megabytes, **mke2fs** will use the file system type *floppy*. If the file system size is greater than or equal to 3 but less than 512 megabytes, **mke2fs(8)** will use the file system type *small*. If the file system size is greater than or equal to 4 terabytes but less than 16 terabytes, **mke2fs(8)** will use the file system type *big*. If the file system size is greater than or equal to 16 terabytes, **mke2fs(8)** will use the file system type *huge*. Otherwise, **mke2fs(8)** will use the default file system type *default*.

-U UUID

Set the universally unique identifier (UUID) of the file system to *UUID*. The format of the UUID is a series of hex digits separated by hyphens, like this: "c1b9d5a2-f162-11cf-9ece-0020afc76f16". The *UUID* parameter may also be one of the following:

<i>clear</i>	clear the file system UUID
<i>random</i>	generate a new randomly-generated UUID
<i>time</i>	generate a new time-based UUID

-v Verbose execution.

-V Print the version number of **mke2fs** and exit.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the *undo_file* argument, the undo file will be written to a file named **mke2fs-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable or the *undo_dir* directive in the configuration file.

WARNING: The undo file cannot be used to recover from a power or system crash.

ENVIRONMENT

MKE2FS_SYNC

If set to non-zero integer value, its value is used to determine how often **sync(2)** is called during inode table initialization.

MKE2FS_CONFIG

Determines the location of the configuration file (see **mke2fs.conf(5)**).

MKE2FS_FIRST_META_BG

If set to non-zero integer value, its value is used to determine first meta block group. This is mostly for debugging purposes.

MKE2FS_DEVICE_SECTSIZE

If set to non-zero integer value, its value is used to determine logical sector size of the *device*.

MKE2FS_DEVICE_PHYS_SECTSIZE

If set to non-zero integer value, its value is used to determine physical sector size of the *device*.

MKE2FS_SKIP_CHECK_MSG

If set, do not show the message of file system automatic check caused by mount count or check interval.

AUTHOR

This version of **mke2fs** has been written by Theodore Ts'o <tytso@mit.edu>.

AVAILABILITY

mke2fs is part of the e2fsprogs package and is available from <http://e2fsprogs.sourceforge.net>.

SEE ALSO

mke2fs.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **e2fsck(8)**, **tune2fs(8)**, **ext4(5)**

NAME

mke2fs – create an ext2/ext3/ext4 file system

SYNOPSIS

```
mke2fs [ -c | -I filename ] [ -b block-size ] [ -C cluster-size ] [ -d root-directory ] [ -D ] [ -g blocks-per-group ] [ -G number-of-groups ] [ -i bytes-per-inode ] [ -I inode-size ] [ -j ] [ -J journal-options ] [ -N number-of-inodes ] [ -n ] [ -m reserved-blocks-percentage ] [ -o creator-os ] [ -O [^]feature[,...] ] [ -q ] [ -r fs-revision-level ] [ -E extended-options ] [ -v ] [ -F ] [ -L volume-label ] [ -M last-mounted-directory ] [ -S ] [ -t fs-type ] [ -T usage-type ] [ -U UUID ] [ -V ] [ -e errors-behavior ] [ -z undo_file ] device [ fs-size ]
```

```
mke2fs -O journal_dev [ -b block-size ] [ -L volume-label ] [ -n ] [ -q ] [ -v ] external-journal [ fs-size ]
```

DESCRIPTION

mke2fs is used to create an ext2, ext3, or ext4 file system, usually in a disk partition (or file) named by *device*.

The file system size is specified by *fs-size*. If *fs-size* does not have a suffix, it is interpreted as power-of-two kilobytes, unless the **-b** *blocksize* option is specified, in which case *fs-size* is interpreted as the number of *blocksize* blocks. If the *fs-size* is suffixed by 'k', 'm', 'g', 't' (either upper-case or lower-case), then it is interpreted in power-of-two kilobytes, megabytes, gigabytes, terabytes, etc. If *fs-size* is omitted, **mke2fs** will create the file system based on the device size.

If **mke2fs** is run as **mkfs.XXX** (i.e., **mkfs.ext2**, **mkfs.ext3**, or **mkfs.ext4**) the option **-t XXX** is implied; so **mkfs.ext3** will create a file system for use with ext3, **mkfs.ext4** will create a file system for use with ext4, and so on.

The defaults of the parameters for the newly created file system, if not overridden by the options listed below, are controlled by the */etc/mke2fs.conf* configuration file. See the **mke2fs.conf(5)** manual page for more details.

OPTIONS**-b** *block-size*

Specify the size of blocks in bytes. Valid block-size values are powers of two from 1024 up to 65536 (however note that the kernel is able to mount only file systems with block-size smaller or equal to the system page size - 4k on x86 systems, up to 64k on ppc64 or aarch64 depending on kernel configuration). If omitted, block-size is heuristically determined by the file system size and the expected usage of the file system (see the **-T** option). In most common cases, the default block size is 4k. If *block-size* is preceded by a negative sign ('-'), then **mke2fs** will use heuristics to determine the appropriate block size, with the constraint that the block size will be at least *block-size* bytes. This is useful for certain hardware devices which require that the *blocksize* be a multiple of 2k.

-c Check the device for bad blocks before creating the file system. If this option is specified twice, then a slower read-write test is used instead of a fast read-only test.

-C *cluster-size*

Specify the size of cluster in bytes for file systems using the bigalloc feature. Valid cluster-size values are from 2048 to 256M bytes per cluster. This can only be specified if the bigalloc feature is enabled. (See the **ext4 (5)** man page for more details about bigalloc.) The default cluster size if bigalloc is enabled is 16 times the block size.

-d *root-directory*

Copy the contents of the given directory into the root directory of the file system.

-D Use direct I/O when writing to the disk. This avoids mke2fs dirtying a lot of buffer cache memory, which may impact other applications running on a busy server. This option will cause mke2fs to run much more slowly, however, so there is a tradeoff to using direct I/O.

-e error-behavior

Change the behavior of the kernel code when errors are detected. In all cases, a file system error will cause **e2fsck(8)** to check the file system on the next boot. *error-behavior* can be one of the following:

continue	Continue normal execution.
remount-ro	Remount file system read-only.
panic	Cause a kernel panic.

-E extended-options

Set extended options for the file system. Extended options are comma separated, and may take an argument using the equals ('=') sign. The **-E** option used to be **-R** in earlier versions of **mke2fs**. The **-R** option is still accepted for backwards compatibility, but is deprecated. The following extended options are supported:

encoding=encoding-name

Enable the *casemap* feature in the super block and set *encoding-name* as the encoding to be used. If *encoding-name* is not specified, the encoding defined in **mke2fs.conf(5)** is used.

encoding_flags=encoding-flags

Define parameters for file name character encoding operations. If a flag is not changed using this parameter, its default value is used. *encoding-flags* should be a comma-separated lists of flags to be enabled. To disable a flag, add it to the list with the prefix "no".

The only flag that can be set right now is *strict* which means that invalid strings should be rejected by the file system. In the default configuration, the *strict* flag is disabled.

mmp_update_interval=interval

Adjust the initial MMP update interval to *interval* seconds. Specifying an *interval* of 0 means to use the default interval. The specified interval must be less than 300 seconds. Requires that the **mmp** feature be enabled.

stride=stride-size

Configure the file system for a RAID array with *stride-size* file system blocks. This is the number of blocks read or written to disk before moving to the next disk, which is sometimes referred to as the *chunk size*. This mostly affects placement of file system metadata like bitmaps at **mke2fs** time to avoid placing them on a single disk, which can hurt performance. It may also be used by the block allocator.

stripe_width=stripe-width

Configure the file system for a RAID array with *stripe-width* file system blocks per stripe. This is typically *stride-size* * N, where N is the number of data-bearing disks in the RAID (e.g. for RAID 5 there is one parity disk, so N will be the number of disks in the array minus 1). This allows the block allocator to prevent read-modify-write of the parity in a RAID stripe if possible when the data is written.

offset=offset

Create the file system at an offset from the beginning of the device or file. This can be useful when creating disk images for virtual machines.

resize=max-online-resize

Reserve enough space so that the block group descriptor table can grow to support a file system that has *max-online-resize* blocks.

lazy_itable_init[= <0 to disable, 1 to enable>]

If enabled and the uninit_bg feature is enabled, the inode table will not be fully initialized by **mke2fs**. This speeds up file system initialization noticeably, but it requires the kernel to finish initializing the file system in the background when the file system is first mounted. If the option value is omitted, it defaults to 1 to enable lazy inode table zeroing.

lazy_journal_init[= <0 to disable, 1 to enable>]

If enabled, the journal inode will not be fully zeroed out by **mke2fs**. This speeds up file system initialization noticeably, but carries some small risk if the system crashes before the journal has been overwritten entirely one time. If the option value is omitted, it defaults to 1 to enable lazy journal inode zeroing.

no_copy_xattrs

Normally **mke2fs** will copy the extended attributes of the files in the directory hierarchy specified via the (optional) **-d** option. This will disable the copy and leaves the files in the newly created file system without any extended attributes.

num_backup_sb=<0/1/2>

If the **sparse_super2** file system feature is enabled this option controls whether there will be 0, 1, or 2 backup superblocks created in the file system.

packed_meta_blocks[= <0 to disable, 1 to enable>]

Place the allocation bitmaps and the inode table at the beginning of the disk. This option requires that the flex_bg file system feature to be enabled in order for it to have effect, and will also create the journal at the beginning of the file system. This option is useful for flash devices that use SLC flash at the beginning of the disk. It also maximizes the range of contiguous data blocks, which can be useful for certain specialized use cases, such as supported Shingled Drives.

root_owner[=uid:gid]

Specify the numeric user and group ID of the root directory. If no UID:GID is specified, use the user and group ID of the user running **mke2fs**. In **mke2fs** 1.42 and earlier the UID and GID of the root directory were set by default to the UID and GID of the user running the mke2fs command. The **root_owner=** option allows explicitly specifying these values, and avoid side-effects for users that do not expect the contents of the file system to change based on the user running **mke2fs**.

test_fs Set a flag in the file system superblock indicating that it may be mounted using experimental kernel code, such as the ext4dev file system.

discard

Attempt to discard blocks at mkfs time (discarding blocks initially is useful on solid state devices and sparse / thin-provisioned storage). When the device advertises that discard also zeroes data (any subsequent read after the discard and before write returns zero), then mark all not-yet-zeroed inode tables as zeroed. This significantly speeds up file system initialization. This is set as default.

nodiscard

Do not attempt to discard blocks at mkfs time.

quotatype

Specify the which quota types (usrquota, grpquota, prjquota) which should be enabled in the created file system. The argument of this extended option should be a colon separated list. This option has effect only if the **quota**

feature is set. The default quota types to be initialized if this option is not specified is both user and group quotas. If the project feature is enabled that project quotas will be initialized as well.

- F** Force **mke2fs** to create a file system, even if the specified device is not a partition on a block special device, or if other parameters do not make sense. In order to force **mke2fs** to create a file system even if the file system appears to be in use or is mounted (a truly dangerous thing to do), this option must be specified twice.

-g *blocks-per-group*

Specify the number of blocks in a block group. There is generally no reason for the user to ever set this parameter, as the default is optimal for the file system. (For administrators who are creating file systems on RAID arrays, it is preferable to use the *stride* RAID parameter as part of the **-E** option rather than manipulating the number of blocks per group.) This option is generally used by developers who are developing test cases.

If the bigalloc feature is enabled, the **-g** option will specify the number of clusters in a block group.

-G *number-of-groups*

Specify the number of block groups that will be packed together to create a larger virtual block group (or "flex_bg group") in an ext4 file system. This improves meta-data locality and performance on meta-data heavy workloads. The number of groups must be a power of 2 and may only be specified if the **flex_bg** file system feature is enabled.

-i *bytes-per-inode*

Specify the bytes/inode ratio. **mk e2fs** creates an inode for every *bytes-per-inode* bytes of space on the disk. The larger the *bytes-per-inode* ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the file system, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to change this ratio on a file system after it is created, so be careful deciding the correct value for this parameter. Note that resizing a file system changes the number of inodes to maintain this ratio.

-I *inode-size*

Specify the size of each inode in bytes. The *inode-size* value must be a power of 2 larger or equal to 128. The larger the *inode-size* the more space the inode table will consume, and this reduces the usable space in the file system and can also negatively impact performance. It is not possible to change this value after the file system is created.

File systems with an inode size of 128 bytes do not support timestamps beyond January 19, 2038. Inodes which are 256 bytes or larger will support extended timestamps, project id's, and the ability to store some extended attributes in the inode table for improved performance.

The default inode size is controlled by the **mke2fs.conf(5)** file. In the **mke2fs.conf** file shipped with e2fsprogs, the default inode size is 256 bytes for most file systems, except for small file systems where the inode size will be 128 bytes.

- j** Create the file system with an ext3 journal. If the **-J** option is not specified, the default journal parameters will be used to create an appropriately sized journal (given the size of the file system) stored within the file system. Note that you must be using a kernel which has ext3 support in order to actually make use of the journal.

-J *journal-options*

Create the ext3 journal using options specified on the command-line. Journal options are comma separated, and may take an argument using the equals ('=') sign. The following journal options are supported:

size=*journal-size*

Create an internal journal (i.e., stored inside the file system) of size *journal-size* megabytes. The size of the journal must be at least 1024 file system blocks (i.e., 1MB if using 1k blocks, 4MB if using 4k blocks, etc.) and may

be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller)

fast_commit_size=*fast-commit-size*

Create an additional fast commit journal area of size *fast-commit-size* kilobytes. This option is only valid if **fast_commit** feature is enabled on the file system. If this option is not specified and if **fast_commit** feature is turned on, fast commit area size defaults to *journal-size* / 64 me gabytes. The total size of the journal with **fast_commit** feature set is *journal-size* + (*fast-commit-size* * 1024) megabytes. The total journal size may be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller).

location=*journal-location*

Specify the location of the journal. The argument *journal-location* can either be specified as a block number, or if the number has a units suffix (e.g., 'M', 'G', etc.) interpret it as the offset from the beginning of the file system.

device=*external-journal*

Attach the file system to the journal block device located on *external-journal*. The external journal must already have been created using the command

mke2fs -O journal_dev *external-journal*

Note that *external-journal* must have been created with the same block size as the new file system. In addition, while there is support for attaching multiple file systems to a single external journal, the Linux kernel and **e2fsck(8)** do not currently support shared external journals yet.

Instead of specifying a device name directly, *external-journal* can also be specified by either **LABEL=***label* or **UUID=***UUID* to locate the external journal by either the volume label or UUID stored in the ext2 superblock at the start of the journal. Use **dumpe2fs(8)** to display a journal device's volume label and UUID. See also the **-L** option of **tune2fs(8)**.

Only one of the **size** or **device** options can be given for a file system.

-l *filename*

Read the bad blocks list from *filename*. Note that the block numbers in the bad block list must be generated using the same block size as used by **mke2fs**. As a result, the **-c** option to **mk e2fs** is a much simpler and less error-prone method of checking a disk for bad blocks before formatting it, as **mke2fs** will automatically pass the correct parameters to the **badblocks** program.

-L *new-volume-label*

Set the volume label for the file system to *new-volume-label*. The maximum length of the volume label is 16 bytes.

-m *reserved-blocks-percentage*

Specify the percentage of the file system blocks reserved for the super-user. This avoids fragmentation, and allows root-owned daemons, such as **syslogd(8)**, to continue to function correctly after non-privileged processes are prevented from writing to the file system. The default percentage is 5%.

-M *last-mounted-directory*

Set the last mounted directory for the file system. This might be useful for the sake of utilities that key off of the last mounted directory to determine where the file system should be mounted.

- n** Causes **mke2fs** to not actually create a file system, but display what it would do if it were to create a file system. This can be used to determine the location of the backup superblocks for a particular file system, so long as the **mke2fs** parameters that were passed when the file system was originally created are used again. (With the **-n** option added, of course!)

-N number-of-inodes

Overrides the default calculation of the number of inodes that should be reserved for the file system (which is based on the number of blocks and the *bytes-per-inode* ratio). This allows the user to specify the number of desired inodes directly.

-o creator-os

Overrides the default value of the "creator operating system" field of the file system. The creator field is set by default to the name of the OS the **mke2fs** executable was compiled for.

-O []feature[,...]

Create a file system with the given features (file system options), overriding the default file system options. The features that are enabled by default are specified by the *base_features* relation, either in the *[defaults]* section in the **/etc/mke2fs.conf** configuration file, or in the *[fs_types]* subsections for the usage types as specified by the **-T** option, further modified by the *features* relation found in the *[fs_types]* subsections for the file system and usage types. See the **mke2fs.conf(5)** manual page for more details. The file system type-specific configuration setting found in the *[fs_types]* section will override the global default found in *[defaults]*.

The file system feature set will be further edited using either the feature set specified by this option, or if this option is not given, by the *default_features* relation for the file system type being created, or in the *[defaults]* section of the configuration file.

The file system feature set is comprised of a list of features, separated by commas, that are to be enabled. To disable a feature, simply prefix the feature name with a caret (^) character. Features with dependencies will not be removed successfully. The pseudo-file system feature "none" will clear all file system features.

For more information about the features which can be set, please see the manual page **ext4(5)**.

-q Quiet execution. Useful if **mke2fs** is run in a script.**-r revision**

Set the file system revision for the new file system. Note that 1.2 kernels only support revision 0 file systems. The default is to create revision 1 file systems.

-S

Write superblock and group descriptors only. This is an extreme measure to be taken only in the very unlikely case that all of the superblock and backup superblocks are corrupted, and a last-ditch recovery method is desired by experienced users. It causes **mke2fs** to reinitialize the superblock and group descriptors, while not touching the inode table and the block and inode bitmaps. The **e2fsck** program should be run immediately after this option is used, and there is no guarantee that any data will be salvageable. Due to the wide variety of possible options to **mke2fs** that affect the on-disk layout, it is critical to specify exactly the same format options, such as blocksize, fs-type, feature flags, and other tunables when using this option, or the file system will be further corrupted. In some cases, such as file systems that have been resized, or have had features enabled after format time, it is impossible to overwrite all of the superblocks correctly, and at least some file system corruption will occur. It is best to run this on a full copy of the file system so other options can be tried if this doesn't work.

-t fs-type

Specify the file system type (i.e., ext2, ext3, ext4, etc.) that is to be created. If this option is not specified, **mke2fs** will pick a default either via how the command was run (for example, using a name of the form **mkfs.ext2**, **mkfs.ext3**, etc.) or via a default as defined by the **/etc/mke2fs.conf** file. This option controls which file system options are used by default, based on the **fstypes** configuration stanza in **/etc/mke2fs.conf**.

If the **-O** option is used to explicitly add or remove file system options that should be set in the newly created file system, the resulting file system may not be supported by the requested *fs-type*. (e.g., "**mke2fs -t ext3 -O extent /dev/sdXX**" will create a file system that is not supported by the

ext3 implementation as found in the Linux kernel; and "**mke2fs -t ext3 -O ^has_journal /dev/hdXX**" will create a file system that does not have a journal and hence will not be supported by the ext3 file system code in the Linux kernel.)

-T usage-type[,...]

Specify how the file system is going to be used, so that **mke2fs** can choose optimal file system parameters for that use. The usage types that are supported are defined in the configuration file **/etc/mke2fs.conf**. The user may specify one or more usage types using a comma separated list.

If this option is not specified, **mke2fs** will pick a single default usage type based on the size of the file system to be created. If the file system size is less than 3 megabytes, **mke2fs** will use the file system type *floppy*. If the file system size is greater than or equal to 3 but less than 512 megabytes, **mke2fs(8)** will use the file system type *small*. If the file system size is greater than or equal to 4 terabytes but less than 16 terabytes, **mke2fs(8)** will use the file system type *big*. If the file system size is greater than or equal to 16 terabytes, **mke2fs(8)** will use the file system type *huge*. Otherwise, **mke2fs(8)** will use the default file system type *default*.

-U UUID

Set the universally unique identifier (UUID) of the file system to *UUID*. The format of the UUID is a series of hex digits separated by hyphens, like this: "c1b9d5a2-f162-11cf-9ece-0020afc76f16". The *UUID* parameter may also be one of the following:

<i>clear</i>	clear the file system UUID
<i>random</i>	generate a new randomly-generated UUID
<i>time</i>	generate a new time-based UUID

-v Verbose execution.

-V Print the version number of **mke2fs** and exit.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the *undo_file* argument, the undo file will be written to a file named **mke2fs-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable or the *undo_dir* directive in the configuration file.

WARNING: The undo file cannot be used to recover from a power or system crash.

ENVIRONMENT

MKE2FS_SYNC

If set to non-zero integer value, its value is used to determine how often **sync(2)** is called during inode table initialization.

MKE2FS_CONFIG

Determines the location of the configuration file (see **mke2fs.conf(5)**).

MKE2FS_FIRST_META_BG

If set to non-zero integer value, its value is used to determine first meta block group. This is mostly for debugging purposes.

MKE2FS_DEVICE_SECTSIZE

If set to non-zero integer value, its value is used to determine logical sector size of the *device*.

MKE2FS_DEVICE_PHYS_SECTSIZE

If set to non-zero integer value, its value is used to determine physical sector size of the *device*.

MKE2FS_SKIP_CHECK_MSG

If set, do not show the message of file system automatic check caused by mount count or check interval.

AUTHOR

This version of **mke2fs** has been written by Theodore Ts'o <tytso@mit.edu>.

AVAILABILITY

mke2fs is part of the e2fsprogs package and is available from <http://e2fsprogs.sourceforge.net>.

SEE ALSO

mke2fs.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **e2fsck(8)**, **tune2fs(8)**, **ext4(5)**

NAME

mke2fs – create an ext2/ext3/ext4 file system

SYNOPSIS

```
mke2fs [ -c | -I filename ] [ -b block-size ] [ -C cluster-size ] [ -d root-directory ] [ -D ] [ -g blocks-per-group ] [ -G number-of-groups ] [ -i bytes-per-inode ] [ -I inode-size ] [ -j ] [ -J journal-options ] [ -N number-of-inodes ] [ -n ] [ -m reserved-blocks-percentage ] [ -o creator-os ] [ -O [^]feature[,...] ] [ -q ] [ -r fs-revision-level ] [ -E extended-options ] [ -v ] [ -F ] [ -L volume-label ] [ -M last-mounted-directory ] [ -S ] [ -t fs-type ] [ -T usage-type ] [ -U UUID ] [ -V ] [ -e errors-behavior ] [ -z undo_file ] device [ fs-size ]
```

```
mke2fs -O journal_dev [ -b block-size ] [ -L volume-label ] [ -n ] [ -q ] [ -v ] external-journal [ fs-size ]
```

DESCRIPTION

mke2fs is used to create an ext2, ext3, or ext4 file system, usually in a disk partition (or file) named by *device*.

The file system size is specified by *fs-size*. If *fs-size* does not have a suffix, it is interpreted as power-of-two kilobytes, unless the **-b** *blocksize* option is specified, in which case *fs-size* is interpreted as the number of *blocksize* blocks. If the *fs-size* is suffixed by 'k', 'm', 'g', 't' (either upper-case or lower-case), then it is interpreted in power-of-two kilobytes, megabytes, gigabytes, terabytes, etc. If *fs-size* is omitted, **mke2fs** will create the file system based on the device size.

If **mke2fs** is run as **mkfs.XXX** (i.e., **mkfs.ext2**, **mkfs.ext3**, or **mkfs.ext4**) the option **-t XXX** is implied; so **mkfs.ext3** will create a file system for use with ext3, **mkfs.ext4** will create a file system for use with ext4, and so on.

The defaults of the parameters for the newly created file system, if not overridden by the options listed below, are controlled by the */etc/mke2fs.conf* configuration file. See the **mke2fs.conf(5)** manual page for more details.

OPTIONS**-b** *block-size*

Specify the size of blocks in bytes. Valid block-size values are powers of two from 1024 up to 65536 (however note that the kernel is able to mount only file systems with block-size smaller or equal to the system page size - 4k on x86 systems, up to 64k on ppc64 or aarch64 depending on kernel configuration). If omitted, block-size is heuristically determined by the file system size and the expected usage of the file system (see the **-T** option). In most common cases, the default block size is 4k. If *block-size* is preceded by a negative sign ('-'), then **mke2fs** will use heuristics to determine the appropriate block size, with the constraint that the block size will be at least *block-size* bytes. This is useful for certain hardware devices which require that the *blocksize* be a multiple of 2k.

-c Check the device for bad blocks before creating the file system. If this option is specified twice, then a slower read-write test is used instead of a fast read-only test.

-C *cluster-size*

Specify the size of cluster in bytes for file systems using the bigalloc feature. Valid cluster-size values are from 2048 to 256M bytes per cluster. This can only be specified if the bigalloc feature is enabled. (See the **ext4 (5)** man page for more details about bigalloc.) The default cluster size if bigalloc is enabled is 16 times the block size.

-d *root-directory*

Copy the contents of the given directory into the root directory of the file system.

-D Use direct I/O when writing to the disk. This avoids mke2fs dirtying a lot of buffer cache memory, which may impact other applications running on a busy server. This option will cause mke2fs to run much more slowly, however, so there is a tradeoff to using direct I/O.

-e error-behavior

Change the behavior of the kernel code when errors are detected. In all cases, a file system error will cause **e2fsck(8)** to check the file system on the next boot. *error-behavior* can be one of the following:

continue	Continue normal execution.
remount-ro	Remount file system read-only.
panic	Cause a kernel panic.

-E extended-options

Set extended options for the file system. Extended options are comma separated, and may take an argument using the equals ('=') sign. The **-E** option used to be **-R** in earlier versions of **mke2fs**. The **-R** option is still accepted for backwards compatibility, but is deprecated. The following extended options are supported:

encoding=encoding-name

Enable the *casemap* feature in the super block and set *encoding-name* as the encoding to be used. If *encoding-name* is not specified, the encoding defined in **mke2fs.conf(5)** is used.

encoding_flags=encoding-flags

Define parameters for file name character encoding operations. If a flag is not changed using this parameter, its default value is used. *encoding-flags* should be a comma-separated lists of flags to be enabled. To disable a flag, add it to the list with the prefix "no".

The only flag that can be set right now is *strict* which means that invalid strings should be rejected by the file system. In the default configuration, the *strict* flag is disabled.

mmp_update_interval=interval

Adjust the initial MMP update interval to *interval* seconds. Specifying an *interval* of 0 means to use the default interval. The specified interval must be less than 300 seconds. Requires that the **mmp** feature be enabled.

stride=stride-size

Configure the file system for a RAID array with *stride-size* file system blocks. This is the number of blocks read or written to disk before moving to the next disk, which is sometimes referred to as the *chunk size*. This mostly affects placement of file system metadata like bitmaps at **mke2fs** time to avoid placing them on a single disk, which can hurt performance. It may also be used by the block allocator.

stripe_width=stripe-width

Configure the file system for a RAID array with *stripe-width* file system blocks per stripe. This is typically *stride-size* * N, where N is the number of data-bearing disks in the RAID (e.g. for RAID 5 there is one parity disk, so N will be the number of disks in the array minus 1). This allows the block allocator to prevent read-modify-write of the parity in a RAID stripe if possible when the data is written.

offset=offset

Create the file system at an offset from the beginning of the device or file. This can be useful when creating disk images for virtual machines.

resize=max-online-resize

Reserve enough space so that the block group descriptor table can grow to support a file system that has *max-online-resize* blocks.

lazy_itable_init[= <0 to disable, 1 to enable>]

If enabled and the uninit_bg feature is enabled, the inode table will not be fully initialized by **mke2fs**. This speeds up file system initialization noticeably, but it requires the kernel to finish initializing the file system in the background when the file system is first mounted. If the option value is omitted, it defaults to 1 to enable lazy inode table zeroing.

lazy_journal_init[= <0 to disable, 1 to enable>]

If enabled, the journal inode will not be fully zeroed out by **mke2fs**. This speeds up file system initialization noticeably, but carries some small risk if the system crashes before the journal has been overwritten entirely one time. If the option value is omitted, it defaults to 1 to enable lazy journal inode zeroing.

no_copy_xattrs

Normally **mke2fs** will copy the extended attributes of the files in the directory hierarchy specified via the (optional) **-d** option. This will disable the copy and leaves the files in the newly created file system without any extended attributes.

num_backup_sb=<0/1/2>

If the **sparse_super2** file system feature is enabled this option controls whether there will be 0, 1, or 2 backup superblocks created in the file system.

packed_meta_blocks[= <0 to disable, 1 to enable>]

Place the allocation bitmaps and the inode table at the beginning of the disk. This option requires that the flex_bg file system feature to be enabled in order for it to have effect, and will also create the journal at the beginning of the file system. This option is useful for flash devices that use SLC flash at the beginning of the disk. It also maximizes the range of contiguous data blocks, which can be useful for certain specialized use cases, such as supported Shingled Drives.

root_owner[=uid:gid]

Specify the numeric user and group ID of the root directory. If no UID:GID is specified, use the user and group ID of the user running **mke2fs**. In **mke2fs** 1.42 and earlier the UID and GID of the root directory were set by default to the UID and GID of the user running the mke2fs command. The **root_owner=** option allows explicitly specifying these values, and avoid side-effects for users that do not expect the contents of the file system to change based on the user running **mke2fs**.

test_fs Set a flag in the file system superblock indicating that it may be mounted using experimental kernel code, such as the ext4dev file system.

discard

Attempt to discard blocks at mkfs time (discarding blocks initially is useful on solid state devices and sparse / thin-provisioned storage). When the device advertises that discard also zeroes data (any subsequent read after the discard and before write returns zero), then mark all not-yet-zeroed inode tables as zeroed. This significantly speeds up file system initialization. This is set as default.

nodiscard

Do not attempt to discard blocks at mkfs time.

quotatype

Specify the which quota types (usrquota, grpquota, prjquota) which should be enabled in the created file system. The argument of this extended option should be a colon separated list. This option has effect only if the **quota**

feature is set. The default quota types to be initialized if this option is not specified is both user and group quotas. If the project feature is enabled that project quotas will be initialized as well.

- F** Force **mke2fs** to create a file system, even if the specified device is not a partition on a block special device, or if other parameters do not make sense. In order to force **mke2fs** to create a file system even if the file system appears to be in use or is mounted (a truly dangerous thing to do), this option must be specified twice.

-g *blocks-per-group*

Specify the number of blocks in a block group. There is generally no reason for the user to ever set this parameter, as the default is optimal for the file system. (For administrators who are creating file systems on RAID arrays, it is preferable to use the *stride* RAID parameter as part of the **-E** option rather than manipulating the number of blocks per group.) This option is generally used by developers who are developing test cases.

If the bigalloc feature is enabled, the **-g** option will specify the number of clusters in a block group.

-G *number-of-groups*

Specify the number of block groups that will be packed together to create a larger virtual block group (or "flex_bg group") in an ext4 file system. This improves meta-data locality and performance on meta-data heavy workloads. The number of groups must be a power of 2 and may only be specified if the **flex_bg** file system feature is enabled.

-i *bytes-per-inode*

Specify the bytes/inode ratio. **mk e2fs** creates an inode for every *bytes-per-inode* bytes of space on the disk. The larger the *bytes-per-inode* ratio, the fewer inodes will be created. This value generally shouldn't be smaller than the blocksize of the file system, since in that case more inodes would be made than can ever be used. Be warned that it is not possible to change this ratio on a file system after it is created, so be careful deciding the correct value for this parameter. Note that resizing a file system changes the number of inodes to maintain this ratio.

-I *inode-size*

Specify the size of each inode in bytes. The *inode-size* value must be a power of 2 larger or equal to 128. The larger the *inode-size* the more space the inode table will consume, and this reduces the usable space in the file system and can also negatively impact performance. It is not possible to change this value after the file system is created.

File systems with an inode size of 128 bytes do not support timestamps beyond January 19, 2038. Inodes which are 256 bytes or larger will support extended timestamps, project id's, and the ability to store some extended attributes in the inode table for improved performance.

The default inode size is controlled by the **mke2fs.conf(5)** file. In the **mke2fs.conf** file shipped with e2fsprogs, the default inode size is 256 bytes for most file systems, except for small file systems where the inode size will be 128 bytes.

- j** Create the file system with an ext3 journal. If the **-J** option is not specified, the default journal parameters will be used to create an appropriately sized journal (given the size of the file system) stored within the file system. Note that you must be using a kernel which has ext3 support in order to actually make use of the journal.

-J *journal-options*

Create the ext3 journal using options specified on the command-line. Journal options are comma separated, and may take an argument using the equals ('=') sign. The following journal options are supported:

size=*journal-size*

Create an internal journal (i.e., stored inside the file system) of size *journal-size* megabytes. The size of the journal must be at least 1024 file system blocks (i.e., 1MB if using 1k blocks, 4MB if using 4k blocks, etc.) and may

be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller)

fast_commit_size=*fast-commit-size*

Create an additional fast commit journal area of size *fast-commit-size* kilobytes. This option is only valid if **fast_commit** feature is enabled on the file system. If this option is not specified and if **fast_commit** feature is turned on, fast commit area size defaults to *journal-size* / 64 me gabytes. The total size of the journal with **fast_commit** feature set is *journal-size* + (*fast-commit-size* * 1024) megabytes. The total journal size may be no more than 10,240,000 file system blocks or half the total file system size (whichever is smaller).

location=*journal-location*

Specify the location of the journal. The argument *journal-location* can either be specified as a block number, or if the number has a units suffix (e.g., 'M', 'G', etc.) interpret it as the offset from the beginning of the file system.

device=*external-journal*

Attach the file system to the journal block device located on *external-journal*. The external journal must already have been created using the command

mke2fs -O journal_dev *external-journal*

Note that *external-journal* must have been created with the same block size as the new file system. In addition, while there is support for attaching multiple file systems to a single external journal, the Linux kernel and **e2fsck(8)** do not currently support shared external journals yet.

Instead of specifying a device name directly, *external-journal* can also be specified by either **LABEL=***label* or **UUID=***UUID* to locate the external journal by either the volume label or UUID stored in the ext2 superblock at the start of the journal. Use **dumpe2fs(8)** to display a journal device's volume label and UUID. See also the **-L** option of **tune2fs(8)**.

Only one of the **size** or **device** options can be given for a file system.

-l *filename*

Read the bad blocks list from *filename*. Note that the block numbers in the bad block list must be generated using the same block size as used by **mke2fs**. As a result, the **-c** option to **mk e2fs** is a much simpler and less error-prone method of checking a disk for bad blocks before formatting it, as **mke2fs** will automatically pass the correct parameters to the **badblocks** program.

-L *new-volume-label*

Set the volume label for the file system to *new-volume-label*. The maximum length of the volume label is 16 bytes.

-m *reserved-blocks-percentage*

Specify the percentage of the file system blocks reserved for the super-user. This avoids fragmentation, and allows root-owned daemons, such as **syslogd(8)**, to continue to function correctly after non-privileged processes are prevented from writing to the file system. The default percentage is 5%.

-M *last-mounted-directory*

Set the last mounted directory for the file system. This might be useful for the sake of utilities that key off of the last mounted directory to determine where the file system should be mounted.

- n** Causes **mke2fs** to not actually create a file system, but display what it would do if it were to create a file system. This can be used to determine the location of the backup superblocks for a particular file system, so long as the **mke2fs** parameters that were passed when the file system was originally created are used again. (With the **-n** option added, of course!)

-N number-of-inodes

Overrides the default calculation of the number of inodes that should be reserved for the file system (which is based on the number of blocks and the *bytes-per-inode* ratio). This allows the user to specify the number of desired inodes directly.

-o creator-os

Overrides the default value of the "creator operating system" field of the file system. The creator field is set by default to the name of the OS the **mke2fs** executable was compiled for.

-O []feature[,...]

Create a file system with the given features (file system options), overriding the default file system options. The features that are enabled by default are specified by the *base_features* relation, either in the *[defaults]* section in the **/etc/mke2fs.conf** configuration file, or in the *[fs_types]* subsections for the usage types as specified by the **-T** option, further modified by the *features* relation found in the *[fs_types]* subsections for the file system and usage types. See the **mke2fs.conf(5)** manual page for more details. The file system type-specific configuration setting found in the *[fs_types]* section will override the global default found in *[defaults]*.

The file system feature set will be further edited using either the feature set specified by this option, or if this option is not given, by the *default_features* relation for the file system type being created, or in the *[defaults]* section of the configuration file.

The file system feature set is comprised of a list of features, separated by commas, that are to be enabled. To disable a feature, simply prefix the feature name with a caret (^) character. Features with dependencies will not be removed successfully. The pseudo-file system feature "none" will clear all file system features.

For more information about the features which can be set, please see the manual page **ext4(5)**.

-q Quiet execution. Useful if **mke2fs** is run in a script.**-r revision**

Set the file system revision for the new file system. Note that 1.2 kernels only support revision 0 file systems. The default is to create revision 1 file systems.

-S

Write superblock and group descriptors only. This is an extreme measure to be taken only in the very unlikely case that all of the superblock and backup superblocks are corrupted, and a last-ditch recovery method is desired by experienced users. It causes **mke2fs** to reinitialize the superblock and group descriptors, while not touching the inode table and the block and inode bitmaps. The **e2fsck** program should be run immediately after this option is used, and there is no guarantee that any data will be salvageable. Due to the wide variety of possible options to **mke2fs** that affect the on-disk layout, it is critical to specify exactly the same format options, such as blocksize, fs-type, feature flags, and other tunables when using this option, or the file system will be further corrupted. In some cases, such as file systems that have been resized, or have had features enabled after format time, it is impossible to overwrite all of the superblocks correctly, and at least some file system corruption will occur. It is best to run this on a full copy of the file system so other options can be tried if this doesn't work.

-t fs-type

Specify the file system type (i.e., ext2, ext3, ext4, etc.) that is to be created. If this option is not specified, **mke2fs** will pick a default either via how the command was run (for example, using a name of the form **mkfs.ext2**, **mkfs.ext3**, etc.) or via a default as defined by the **/etc/mke2fs.conf** file. This option controls which file system options are used by default, based on the **fstypes** configuration stanza in **/etc/mke2fs.conf**.

If the **-O** option is used to explicitly add or remove file system options that should be set in the newly created file system, the resulting file system may not be supported by the requested *fs-type*. (e.g., "**mke2fs -t ext3 -O extent /dev/sdXX**" will create a file system that is not supported by the

ext3 implementation as found in the Linux kernel; and "**mke2fs -t ext3 -O ^has_journal /dev/hdXX**" will create a file system that does not have a journal and hence will not be supported by the ext3 file system code in the Linux kernel.)

-T usage-type[,...]

Specify how the file system is going to be used, so that **mke2fs** can choose optimal file system parameters for that use. The usage types that are supported are defined in the configuration file **/etc/mke2fs.conf**. The user may specify one or more usage types using a comma separated list.

If this option is not specified, **mke2fs** will pick a single default usage type based on the size of the file system to be created. If the file system size is less than 3 megabytes, **mke2fs** will use the file system type *floppy*. If the file system size is greater than or equal to 3 but less than 512 megabytes, **mke2fs(8)** will use the file system type *small*. If the file system size is greater than or equal to 4 terabytes but less than 16 terabytes, **mke2fs(8)** will use the file system type *big*. If the file system size is greater than or equal to 16 terabytes, **mke2fs(8)** will use the file system type *huge*. Otherwise, **mke2fs(8)** will use the default file system type *default*.

-U UUID

Set the universally unique identifier (UUID) of the file system to *UUID*. The format of the UUID is a series of hex digits separated by hyphens, like this: "c1b9d5a2-f162-11cf-9ece-0020afc76f16". The *UUID* parameter may also be one of the following:

<i>clear</i>	clear the file system UUID
<i>random</i>	generate a new randomly-generated UUID
<i>time</i>	generate a new time-based UUID

-v Verbose execution.

-V Print the version number of **mke2fs** and exit.

-z undo_file

Before overwriting a file system block, write the old contents of the block to an undo file. This undo file can be used with **e2undo(8)** to restore the old contents of the file system should something go wrong. If the empty string is passed as the *undo_file* argument, the undo file will be written to a file named **mke2fs-device.e2undo** in the directory specified via the **E2FSPROGS_UNDO_DIR** environment variable or the *undo_dir* directive in the configuration file.

WARNING: The undo file cannot be used to recover from a power or system crash.

ENVIRONMENT

MKE2FS_SYNC

If set to non-zero integer value, its value is used to determine how often **sync(2)** is called during inode table initialization.

MKE2FS_CONFIG

Determines the location of the configuration file (see **mke2fs.conf(5)**).

MKE2FS_FIRST_META_BG

If set to non-zero integer value, its value is used to determine first meta block group. This is mostly for debugging purposes.

MKE2FS_DEVICE_SECTSIZE

If set to non-zero integer value, its value is used to determine logical sector size of the *device*.

MKE2FS_DEVICE_PHYS_SECTSIZE

If set to non-zero integer value, its value is used to determine physical sector size of the *device*.

MKE2FS_SKIP_CHECK_MSG

If set, do not show the message of file system automatic check caused by mount count or check interval.

AUTHOR

This version of **mke2fs** has been written by Theodore Ts'o <tytso@mit.edu>.

AVAILABILITY

mke2fs is part of the e2fsprogs package and is available from <http://e2fsprogs.sourceforge.net>.

SEE ALSO

mke2fs.conf(5), **badblocks(8)**, **dumpe2fs(8)**, **e2fsck(8)**, **tune2fs(8)**, **ext4(5)**

NAME

mkfs.f2fs – create an F2FS file system

SYNOPSIS

```
mkfs.f2fs [ -a heap-based-allocation ] [ -c device-list ] [ -d debug-level ] [ -e extension-list ] [ -E extension-list ] [ -f ] [ -g ] [ -i ] [ -l volume-label ] [ -m ] [ -o overprovision-ratio-percentage ] [ -O feature-list ] [ -C encoding:flags ] [ -q ] [ -r ] [ -R root_owner ] [ -s #-of-segments-per-section ] [ -S ] [ -t nodiscard/discard ] [ -T timestamp ] [ -w wanted-sector-size ] [ -z #-of-sections-per-zone ] [ -V ] device [ sectors ]
```

DESCRIPTION

mkfs.f2fs is used to create a f2fs file system (usually in a disk partition). *device* is the special file corresponding to the device (e.g. */dev/sdXX*). *sectors* is optionally given for specifying the filesystem size.

The exit code returned by **mkfs.f2fs** is 0 on success and 1 on failure.

OPTIONS**-a heap-based-allocation**

Specify 1 or 0 to enable/disable heap based block allocation policy. If the value is equal to 1, each of active log areas are initially assigned separately according to the whole volume size. The default value is 1.

-c device-list

Build f2fs with these additional comma separated devices, so that the user can see all the devices as one big volume. Supports up to 7 devices except meta device.

-d debug-level

Specify the level of debugging options. The default number is 0, which shows basic debugging messages.

-e extension-list

Specify a list of file extensions that f2fs will treat as cold files. The data of files with those extensions will be stored in the cold log. The default list includes most of the multimedia file extensions such as jpg, gif, mpeg, mkv, and so on.

-E extension-list

Specify a list of file extensions that f2fs will treat as hot files. The data of files with those extensions will be stored in the hot log. The default list includes database file extensions, such as db.

-f Force overwrite when an existing filesystem is detected on the device. By default, **mkfs.f2fs** will not write to the device if it suspects that there is a filesystem or partition table on the device already.

-g Add default Android options.

-i Enable extended node bitmap. **-l volume-label** Specify the volume label to the partition mounted as F2FS.

-m Specify f2fs filesystem to supports the block zoned feature. Without it, the filesystem doesn't support the feature.

-o overprovision-ratio-percentage

Specify the percentage of the volume that will be used as overprovision area. This area is hidden to users, and utilized by F2FS cleaner. If not specified, the best number will be assigned automatically according to the partition size.

-O feature-list

Set additional features for the filesystem. Features are comma separated, and the flag can be repeated. The following features are supported:

encrypt	Enable support for filesystem level encryption.
----------------	---

extra_attr	Enable extra attr feature, required for some of the other features.
-------------------	---

project_quota	Enable project ID tracking. This is used for project quota accounting. Requires extra attr.
inode_checksum	Enable inode checksum. Requires extra attr.
flexible_inline_xattr	Enable flexible inline xattr. Requires extra attr.
quota	Enable quotas.
inode_crttime	Enable inode creation time feature. Requires extra attr.
lost_found	Enable lost+found feature.
verity	Reserved feature.
sb_checksum	Enable superblock checksum.
casemap	Enable casemap support in the filesystem. Optional flags can be passed with -C
compression	Enable support for filesystem level compression. Requires extra attr.

-C *encoding:flags*

Support casemapping with a specific encoding, with optional comma separated flags.

encoding:

utf8 Use UTF-8 for casemapping.

flags:

strict This flag specifies that invalid strings should be rejected by the filesystem. Default is disabled.

-q Quiet mode. With it, mkfs.f2fs does not show any messages, including the basic messages.

-r Sets the checkpointing strand seed to 0.

-R Give root_owner option for initial uid/gid assignment. Default is set by getuid()/getgid(), and assigned by "-R \$uid:\$gid".

-s #*of-segments-per-section*

Specify the number of segments per section. A section consists of multiple consecutive segments, and is the unit of garbage collection. The default number is 1, which means one segment is assigned to a section.

-S Enable sparse mode.

-t 1/0 Specify 1 or 0 to enable or disable discard policy, respectively. The default value is 1.

-T *timestamp*

Set inodes times to a given timestamp. By default, the current time will be used. This behaviour corresponds to the value -1.

-w *wanted-sector-size*

Specify the sector size in bytes. Without it, the sectors will be calculated by device sector size.

-z #*of-sections-per-zone*

Specify the number of sections per zone. A zone consists of multiple sections. F2FS allocates segments for active logs with separated zones as much as possible. The default number is 1, which means a zone consists of one section.

sectors Number of sectors. Default is determined by device size.

-V Print the version number and exit.

AUTHOR

This version of **mkfs.f2fs** has been written by Jaegeuk Kim <jaegeuk.kim@samsung.com>.

AVAILABILITY

mkfs.f2fs is available from <git://git.kernel.org/pub/scm/linux/kernel/git/jaegeuk/f2fs-tools.git>.

SEE ALSO

mkfs(8), **fsck.f2fs(8)**, **dump.f2fs(8)**, **defrag.f2fs(8)**, **resize.f2fs(8)**, **sload.f2fs(8)**.

NAME

mkfs.fat – create an MS-DOS FAT filesystem

SYNOPSIS

mkfs.fat [*OPTIONS*] *DEVICE* [*BLOCK-COUNT*]

DESCRIPTION

mkfs.fat is used to create a FAT filesystem on a device or in an image file. *DEVICE* is the special file corresponding to the device (e.g. `/dev/sdXX`) or the image file (which does not need to exist when the option **-C** is given). *BLOCK-COUNT* is the number of blocks on the device and size of one block is always 1024 bytes, independently of the sector size or the cluster size. Therefore *BLOCK-COUNT* specifies size of filesystem in KiB unit and not in the number of sectors (like for all other **mkfs.fat** options). If omitted, **mkfs.fat** automatically chooses a filesystem size to fill the available space.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

In Atari mode, if not directed otherwise by the user, **mkfs.fat** will always use 2 sectors per cluster, since GEMDOS doesn't like other values very much. It will also obey the maximum number of sectors GEMDOS can handle. Larger filesystems are managed by raising the logical sector size. An Atari-compatible serial number for the filesystem is generated, and a 12 bit FAT is used only for filesystems that have one of the usual floppy sizes (720k, 1.2M, 1.44M, 2.88M), a 16 bit FAT otherwise. This can be overridden with the **-F** option. Some PC-specific boot sector fields aren't written, and a boot message (option **-m**) is ignored.

OPTIONS

- a** Normally, for any filesystem except very small ones, **mkfs.fat** will align all the data structures to cluster size, to make sure that as long as the partition is properly aligned, so will all the data structures in the filesystem. This option disables alignment; this may provide a handful of additional clusters of storage at the expense of a significant performance degradation on RAIDs, flash media or large-sector hard disks.
- A** Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.
- b SECTOR-OF-BACKUP**
Selects the location of the backup boot sector for FAT32. Default depends on number of reserved sectors, but usually is sector 6. If there is a free space available after the backup boot sector then backup of the FAT32 info sector is put after the backup boot sector, usually at sector 7. The backup must be within the range of reserved sectors. Value 0 completely disables creating of backup boot and info FAT32 sectors.
- c** Check the device for bad blocks before creating the filesystem.
- C** Create the file given as *DEVICE* on the command line, and write the to-be-created filesystem to it. This can be used to create the new filesystem in a file instead of on a real device, and to avoid using **dd** in advance to create a file of appropriate size. With this option, the *BLOCK-COUNT* must be given, because otherwise the intended size of the filesystem wouldn't be known. The file created is a sparse file, which actually only contains the meta-data areas (boot sector, FATs, and root directory). The data portions won't be stored on the disk, but the file nevertheless will have the correct size. The resulting file can be copied later to a floppy disk or other device, or mounted through a loop device.

-D DRIVE-NUMBER

Specify the BIOS drive number to be stored in the FAT boot sector. For hard disks and removable media it is usually 0x80–0xFF (0x80 is first hard disk C:, 0x81 is second hard disk D:, ...), for floppy devices or partitions to be used for floppy emulation it is 0x00–0x7F (0x00 is first floppy A:, 0x01 is second floppy B:).

-f NUMBER-OF-FATS

Specify the number of file allocation tables in the filesystem. The default is 2.

-F FAT-SIZE

Specifies the type of file allocation tables used (12, 16 or 32 bit). If nothing is specified, **mkfs.fat** will automatically select between 12, 16 and 32 bit, whatever fits better for the filesystem size.

-g HEADS/SECTORS-PER-TRACK

Specify *HEADS* and *SECTORS-PER-TRACK* numbers which represents disk geometry of *DEVICE*. Both numbers are stored into the FAT boot sector. Number *SECTORS-PER-TRACK* is used also for aligning the total count of FAT sectors. By default disk geometry is read from *DEVICE* itself. If it is not available then *LBA-Assist Translation* and translation table from the *SD Card Part 2 File System Specification* based on total number of disk sectors is used.

-h NUMBER-OF-HIDDEN-SECTORS

Specify the number of so-called *hidden sectors*, as stored in the FAT boot sector: this number represents the beginning sector of the partition containing the file system. Normally this is an offset (in sectors) relative to the start of the disk, although for MBR logical volumes contained in an extended partition of type 0x05 (a non-LBA extended partition), a quirk in the MS-DOS implementation of FAT requires it to be relative to the partition's immediate containing Extended Boot Record. Boot code and other software handling FAT volumes may also rely on this field being set up correctly; most modern FAT implementations will ignore it. By default, if the *DEVICE* is a partition block device, **mkfs.fat** uses the partition offset relative to disk start. Otherwise, **mkfs.fat** assumes zero. Use this option to override this behaviour.

-i VOLUME-ID

Sets the volume ID of the newly created filesystem; *VOLUME-ID* is a 32-bit hexadecimal number (for example, 2e24ec82). The default is a number which depends on the filesystem creation time.

-I

Ignore and disable safety checks. By default **mkfs.fat** refuses to create a filesystem on a device with partitions or virtual mapping. **mkfs.fat** will complain and tell you that it refuses to work. This is different when using MO disks. One doesn't always need partitions on MO disks. The filesystem can go directly to the whole disk. Under other OSes this is known as the *superfloppy* format. This switch will force **mkfs.fat** to work properly.

-l FILENAME

Read the bad blocks list from *FILENAME*.

-m MESSAGE-FILE

Sets the message the user receives on attempts to boot this filesystem without having properly installed an operating system. The message file must not exceed 418 bytes once line feeds have been converted to carriage return-line feed combinations, and tabs have been expanded. If the filename is a hyphen (-), the text is taken from standard input.

-M FAT-MEDIA-TYPE

Specify the media type to be stored in the FAT boot sector. This value is usually 0xF8 for hard disks and is 0xF0 or a value from 0xF9 to 0xFF for floppies or partitions to be used for floppy emulation.

--mbr[=y|yes|n|no|a|auto]

Fill (fake) MBR table with disk signature one partition which starts at sector 0 (includes MBR itself) and spans whole disk device. It is needed only for non-removable disks used on Microsoft Windows systems and only when formatting whole unpartitioned disk. Location of the disk signature and partition table overlaps with the end of the first FAT sector (boot code location), therefore there is no additional space usage. Default is *auto* mode in which **mkfs.fat** put MBR table only for non-removable disks when formatting whole unpartitioned disk.

-n VOLUME-NAME

Sets the volume name (label) of the filesystem. The volume name can be up to 11 characters long. Supplying an empty string, a string consisting only of white space or the string "NO NAME" as *VOLUME-NAME* has the same effect as not giving the **-n** option. The default is no label.

--codepage=PAGE

Use DOS codepage *PAGE* to encode label. By default codepage 850 is used.

-r ROOT-DIR-ENTRIES

Select the minimal number of entries available in the root directory. The default is 112 or 224 for floppies and 512 for hard disks. Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-R NUMBER-OF-RESERVED-SECTORS

Select the minimal number of reserved sectors. With FAT32 format at least 2 reserved sectors are needed, the default is 32. Otherwise the default is 1 (only the boot sector). Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-s SECTORS-PER-CLUSTER

Specify the number of disk sectors per cluster. Must be a power of 2, i.e. 1, 2, 4, 8, ... 128.

-S LOGICAL-SECTOR-SIZE

Specify the number of bytes per logical sector. Must be a power of 2 and greater than or equal to 512, i.e. 512, 1024, 2048, 4096, 8192, 16384, or 32768. Values larger than 4096 are not conforming to the FAT file system specification and may not work everywhere.

-v Verbose execution.**--offset SECTOR**

Write the filesystem at a specific sector into the device file. This is useful for creating a filesystem in a partitioned disk image without having to set up a loop device.

--variant TYPE

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

--help

Display option summary and exit.

--invariant

Use constants for normally randomly generated or time based data such as volume ID and creation time. Multiple runs of **mkfs.fat** on the same device create identical results with this option. Its main purpose is testing **mkfs.fat**.

BUGS

mkfs.fat can not create boot-able filesystems. This isn't as easy as you might think at first glance for various reasons and has been discussed a lot already. **mkfs.fat** simply will not support it ;)

SEE ALSO

fatlabel(8), **fsck.fat(8)**

HOME PAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

mkfs.fat – create an MS-DOS FAT filesystem

SYNOPSIS

mkfs.fat [OPTIONS] DEVICE [BLOCK-COUNT]

DESCRIPTION

mkfs.fat is used to create a FAT filesystem on a device or in an image file. *DEVICE* is the special file corresponding to the device (e.g. `/dev/sdXX`) or the image file (which does not need to exist when the option **-C** is given). *BLOCK-COUNT* is the number of blocks on the device and size of one block is always 1024 bytes, independently of the sector size or the cluster size. Therefore *BLOCK-COUNT* specifies size of filesystem in KiB unit and not in the number of sectors (like for all other **mkfs.fat** options). If omitted, **mkfs.fat** automatically chooses a filesystem size to fill the available space.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

In Atari mode, if not directed otherwise by the user, **mkfs.fat** will always use 2 sectors per cluster, since GEMDOS doesn't like other values very much. It will also obey the maximum number of sectors GEMDOS can handle. Larger filesystems are managed by raising the logical sector size. An Atari-compatible serial number for the filesystem is generated, and a 12 bit FAT is used only for filesystems that have one of the usual floppy sizes (720k, 1.2M, 1.44M, 2.88M), a 16 bit FAT otherwise. This can be overridden with the **-F** option. Some PC-specific boot sector fields aren't written, and a boot message (option **-m**) is ignored.

OPTIONS

- a** Normally, for any filesystem except very small ones, **mkfs.fat** will align all the data structures to cluster size, to make sure that as long as the partition is properly aligned, so will all the data structures in the filesystem. This option disables alignment; this may provide a handful of additional clusters of storage at the expense of a significant performance degradation on RAIDs, flash media or large-sector hard disks.
- A** Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.
- b SECTOR-OF-BACKUP**
Selects the location of the backup boot sector for FAT32. Default depends on number of reserved sectors, but usually is sector 6. If there is a free space available after the backup boot sector then backup of the FAT32 info sector is put after the backup boot sector, usually at sector 7. The backup must be within the range of reserved sectors. Value 0 completely disables creating of backup boot and info FAT32 sectors.
- c** Check the device for bad blocks before creating the filesystem.
- C** Create the file given as *DEVICE* on the command line, and write the to-be-created filesystem to it. This can be used to create the new filesystem in a file instead of on a real device, and to avoid using **dd** in advance to create a file of appropriate size. With this option, the *BLOCK-COUNT* must be given, because otherwise the intended size of the filesystem wouldn't be known. The file created is a sparse file, which actually only contains the meta-data areas (boot sector, FATs, and root directory). The data portions won't be stored on the disk, but the file nevertheless will have the correct size. The resulting file can be copied later to a floppy disk or other device, or mounted through a loop device.
- D DRIVE-NUMBER**
Specify the BIOS drive number to be stored in the FAT boot sector. For hard disks and removable media it is usually 0x80–0xFF (0x80 is first hard disk C:, 0x81 is second hard disk D:, ...), for floppy devices or partitions to be used for floppy emulation it is 0x00–0x7F (0x00 is first floppy A:, 0x01 is second floppy B:).

-f NUMBER-OF-FATS

Specify the number of file allocation tables in the filesystem. The default is 2.

-F FAT-SIZE

Specifies the type of file allocation tables used (12, 16 or 32 bit). If nothing is specified, **mkfs.fat** will automatically select between 12, 16 and 32 bit, whatever fits better for the filesystem size.

-g HEADS/SECTORS-PER-TRACK

Specify *HEADS* and *SECTORS-PER-TRACK* numbers which represents disk geometry of *DEVICE*. Both numbers are stored into the FAT boot sector. Number *SECTORS-PER-TRACK* is used also for aligning the total count of FAT sectors. By default disk geometry is read from *DEVICE* itself. If it is not available then *LBA-Assist Translation* and translation table from the *SD Card Part 2 File System Specification* based on total number of disk sectors is used.

-h NUMBER-OF-HIDDEN-SECTORS

Specify the number of so-called *hidden sectors*, as stored in the FAT boot sector: this number represents the beginning sector of the partition containing the file system. Normally this is an offset (in sectors) relative to the start of the disk, although for MBR logical volumes contained in an extended partition of type 0x05 (a non-LBA extended partition), a quirk in the MS-DOS implementation of FAT requires it to be relative to the partition's immediate containing Extended Boot Record. Boot code and other software handling FAT volumes may also rely on this field being set up correctly; most modern FAT implementations will ignore it. By default, if the *DEVICE* is a partition block device, **mkfs.fat** uses the partition offset relative to disk start. Otherwise, **mkfs.fat** assumes zero. Use this option to override this behaviour.

-i VOLUME-ID

Sets the volume ID of the newly created filesystem; *VOLUME-ID* is a 32-bit hexadecimal number (for example, 2e24ec82). The default is a number which depends on the filesystem creation time.

-I

Ignore and disable safety checks. By default **mkfs.fat** refuses to create a filesystem on a device with partitions or virtual mapping. **mkfs.fat** will complain and tell you that it refuses to work. This is different when using MO disks. One doesn't always need partitions on MO disks. The filesystem can go directly to the whole disk. Under other OSes this is known as the *superfloppy* format. This switch will force **mkfs.fat** to work properly.

-l FILENAME

Read the bad blocks list from *FILENAME*.

-m MESSAGE-FILE

Sets the message the user receives on attempts to boot this filesystem without having properly installed an operating system. The message file must not exceed 418 bytes once line feeds have been converted to carriage return-line feed combinations, and tabs have been expanded. If the filename is a hyphen (-), the text is taken from standard input.

-M FAT-MEDIA-TYPE

Specify the media type to be stored in the FAT boot sector. This value is usually 0xF8 for hard disks and is 0xF0 or a value from 0xF9 to 0xFF for floppies or partitions to be used for floppy emulation.

--mbr[=y|yes|n|no|a|auto]

Fill (fake) MBR table with disk signature one partition which starts at sector 0 (includes MBR itself) and spans whole disk device. It is needed only for non-removable disks used on Microsoft Windows systems and only when formatting whole unpartitioned disk. Location of the disk signature and partition table overlaps with the end of the first FAT sector (boot code location), therefore there is no additional space usage. Default is *auto* mode in which **mkfs.fat** put MBR table only for non-removable disks when formatting whole unpartitioned disk.

-n VOLUME-NAME

Sets the volume name (label) of the filesystem. The volume name can be up to 11 characters long. Supplying an empty string, a string consisting only of white space or the string "NO NAME" as *VOLUME-NAME* has the same effect as not giving the **-n** option. The default is no label.

--codepage=PAGE

Use DOS codepage *PAGE* to encode label. By default codepage 850 is used.

-r ROOT-DIR-ENTRIES

Select the minimal number of entries available in the root directory. The default is 112 or 224 for floppies and 512 for hard disks. Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-R NUMBER-OF-RESERVED-SECTORS

Select the minimal number of reserved sectors. With FAT32 format at least 2 reserved sectors are needed, the default is 32. Otherwise the default is 1 (only the boot sector). Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-s SECTORS-PER-CLUSTER

Specify the number of disk sectors per cluster. Must be a power of 2, i.e. 1, 2, 4, 8, ... 128.

-S LOGICAL-SECTOR-SIZE

Specify the number of bytes per logical sector. Must be a power of 2 and greater than or equal to 512, i.e. 512, 1024, 2048, 4096, 8192, 16384, or 32768. Values larger than 4096 are not conforming to the FAT file system specification and may not work everywhere.

-v Verbose execution.**--offset SECTOR**

Write the filesystem at a specific sector into the device file. This is useful for creating a filesystem in a partitioned disk image without having to set up a loop device.

--variant TYPE

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

--help

Display option summary and exit.

--invariant

Use constants for normally randomly generated or time based data such as volume ID and creation time. Multiple runs of **mkfs.fat** on the same device create identical results with this option. Its main purpose is testing **mkfs.fat**.

BUGS

mkfs.fat can not create boot-able filesystems. This isn't as easy as you might think at first glance for various reasons and has been discussed a lot already. **mkfs.fat** simply will not support it ;)

SEE ALSO

fatlabel(8), **fsck.fat(8)**

HOME PAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

mkntfs – create an NTFS file system

SYNOPSIS

```
mkntfs [options] device [number-of-sectors]  

mkntfs [ -C ] [ -c cluster-size ] [ -F ] [ -f ] [ -H heads ] [ -h ] [ -I ] [ -L volume-label ] [ -l ] [ -n ] [  

-p part-start-sect ] [ -Q ] [ -q ] [ -S sectors-per-track ] [ -s sector-size ] [ -T ] [ -U ] [ -V ] [ -v ] [  

-z mft-zone-multiplier ] [ --debug ] device [number-of-sectors]
```

DESCRIPTION

mkntfs is used to create an NTFS file system on a device (usually a disk partition) or file. *device* is the special file corresponding to the device (e.g. */dev/hdXX*). *number-of-sectors* is the number of sectors on the device. If omitted, **mkntfs** automagically figures the file system size.

OPTIONS

Below is a summary of all the options that **mkntfs** accepts. Nearly all options have two equivalent names. The short name is preceded by **-** and the long name is preceded by **--**. Any single letter options, that don't take an argument, can be combined into a single command, e.g. **-fv** is equivalent to **-f -v**. Long named options can be abbreviated to any unique prefix of their name.

Basic options

-f, --fast, -Q, --quick

Perform quick (fast) format. This will skip both zeroing of the volume and bad sector checking.

-L, --label STRING

Set the volume label for the filesystem.

-C, --enable-compression

Enable compression on the volume.

-n, --no-action

Causes **mkntfs** to not actually create a filesystem, but display what it would do if it were to create a filesystem. All steps of the format are carried out except the actual writing to the device.

Advanced options

-c, --cluster-size BYTES

Specify the size of clusters in bytes. Valid cluster size values are powers of two, with at least 256, and at most 2097152 bytes (2MB) per cluster. If omitted, **mkntfs** uses 4096 bytes as the default cluster size.

Note that the default cluster size is set to be at least equal to the sector size as a cluster cannot be smaller than a sector. Also, note that values greater than 4096 have the side effect that compression is disabled on the volume (due to limitations in the NTFS compression algorithm currently in use by Windows).

-s, --sector-size BYTES

Specify the size of sectors in bytes. Valid sector size values are 256, 512, 1024, 2048 and 4096 bytes per sector. If omitted, **mkntfs** attempts to determine the *sector-size* automatically and if that fails a default of 512 bytes per sector is used.

-p, --partition-start SECTOR

Specify the partition start sector. The maximum is 4294967295 ($2^{32}-1$). If omitted, **mkntfs** attempts to determine *part-start-sect* automatically and if that fails or the value is oversized, a default of 0 is used. The partition is usable despite a wrong value, however note that a correct *part-start-sect* is required for Windows to be able to boot from the created volume.

-H, --heads NUM

Specify the number of heads. The maximum is 65535 (0xffff). If omitted, **mkntfs** attempts to determine the number of *heads* automatically and if that fails a default of 0 is used. Note that *heads* is required for Windows to be able to boot from the created volume.

-S, --sectors-per-track NUM

Specify the number of sectors per track. The maximum is 65535 (0xffff). If omitted, **mkntfs** attempts to determine the number of *sectors-per-track* automatically and if that fails a default of 0 is used. Note that *sectors-per-track* is required for Windows to be able to boot from the created volume.

-z, --mft-zone-multiplier NUM

Set the MFT zone multiplier, which determines the size of the MFT zone to use on the volume. The MFT zone is the area at the beginning of the volume reserved for the master file table (MFT), which stores the on disk inodes (MFT records). It is noteworthy that small files are stored entirely within the inode; thus, if you expect to use the volume for storing large numbers of very small files, it is useful to set the zone multiplier to a higher value. Note, that the MFT zone is resized on the fly as required during operation of the NTFS driver but choosing a good value will reduce fragmentation. Valid values are 1, 2, 3 and 4. The values have the following meaning:

MFT zone multiplier	MFT zone size (% of volume size)
1	12.5% (default)
2	25.0%
3	37.5%
4	50.0%

-T, --zero-time

Fake the time to be 00:00:00 UTC, Jan 1, 1970 instead of the current system time. This is only really useful for debugging purposes.

-U, --with-uuid

Generate a random volume UUID.

-I, --no-indexing

Disable content indexing on the volume. (This is only meaningful on Windows 2000 and later. Windows NT 4.0 and earlier ignore this as they do not implement content indexing at all.)

-F, --force

Force **mkntfs** to run, even if the specified *device* is not a block special device, or appears to be mounted.

Output options**-q, --quiet**

Quiet execution; only errors are written to stderr, no output to stdout occurs at all. Useful if **mkntfs** is run in a script.

-v, --verbose

Verbose execution.

--debug

Really verbose execution; includes the verbose output from the **-v** option as well as additional output useful for debugging **mkntfs**.

Help options**-V, --version**

Print the version number of **mkntfs** and exit.

-l, --license

Print the licensing information of **mkntfs** and exit.

-h, --help

Show a list of options with a brief description of each one.

KNOWN ISSUES

When applying chkdsk to a file system, it sometimes throws a warning "Correcting errors in the uppercase file." The uppercase file is created while formatting and it defines the mapping of lower case characters to upper case ones, as needed to sort file names in directories. The warning means that the uppercase file defined on the file system is not the same as the one used by the Windows OS on which chkdsk is running, and this may happen because newer versions of Windows take into account new characters defined by the Unicode consortium.

Currently, mkntfs creates the uppercase table so that no warning is thrown by Windows Vista, Windows 7 or Windows 8. A warning may be thrown by other Windows versions, or if chkdsk is applied in succession on different Windows versions.

BUGS

If you find a bug please send an email describing the problem to the development team:
ntfs-3g-devel@lists.sf.net

AUTHORS

mkntfs was written by Anton Altaparmakov, Richard Russon, Erik Sornes and Szabolcs Szakacsits. It was ported to ntfs-3g by Erik Larsson and Jean-Pierre Andre.

AVAILABILITY

mkntfs is part of the **ntfs-3g** package and is available from:
<https://github.com/tuxera/ntfs-3g/wiki/>

SEE ALSO

badblocks(8), **ntfsprogs(8)**

NAME

mkfs.fat – create an MS-DOS FAT filesystem

SYNOPSIS

mkfs.fat [OPTIONS] DEVICE [BLOCK-COUNT]

DESCRIPTION

mkfs.fat is used to create a FAT filesystem on a device or in an image file. *DEVICE* is the special file corresponding to the device (e.g. `/dev/sdXX`) or the image file (which does not need to exist when the option **-C** is given). *BLOCK-COUNT* is the number of blocks on the device and size of one block is always 1024 bytes, independently of the sector size or the cluster size. Therefore *BLOCK-COUNT* specifies size of filesystem in KiB unit and not in the number of sectors (like for all other **mkfs.fat** options). If omitted, **mkfs.fat** automatically chooses a filesystem size to fill the available space.

Two different variants of the FAT filesystem are supported. Standard is the FAT12, FAT16 and FAT32 filesystems as defined by Microsoft and widely used on hard disks and removable media like USB sticks and SD cards. The other is the legacy Atari variant used on Atari ST.

In Atari mode, if not directed otherwise by the user, **mkfs.fat** will always use 2 sectors per cluster, since GEMDOS doesn't like other values very much. It will also obey the maximum number of sectors GEMDOS can handle. Larger filesystems are managed by raising the logical sector size. An Atari-compatible serial number for the filesystem is generated, and a 12 bit FAT is used only for filesystems that have one of the usual floppy sizes (720k, 1.2M, 1.44M, 2.88M), a 16 bit FAT otherwise. This can be overridden with the **-F** option. Some PC-specific boot sector fields aren't written, and a boot message (option **-m**) is ignored.

OPTIONS

- a** Normally, for any filesystem except very small ones, **mkfs.fat** will align all the data structures to cluster size, to make sure that as long as the partition is properly aligned, so will all the data structures in the filesystem. This option disables alignment; this may provide a handful of additional clusters of storage at the expense of a significant performance degradation on RAIDs, flash media or large-sector hard disks.
- A** Select using the Atari variation of the FAT filesystem if that isn't active already, otherwise select standard FAT filesystem. This is selected by default if **mkfs.fat** is run on 68k Atari Linux.
- b SECTOR-OF-BACKUP**
Selects the location of the backup boot sector for FAT32. Default depends on number of reserved sectors, but usually is sector 6. If there is a free space available after the backup boot sector then backup of the FAT32 info sector is put after the backup boot sector, usually at sector 7. The backup must be within the range of reserved sectors. Value 0 completely disables creating of backup boot and info FAT32 sectors.
- c** Check the device for bad blocks before creating the filesystem.
- C** Create the file given as *DEVICE* on the command line, and write the to-be-created filesystem to it. This can be used to create the new filesystem in a file instead of on a real device, and to avoid using **dd** in advance to create a file of appropriate size. With this option, the *BLOCK-COUNT* must be given, because otherwise the intended size of the filesystem wouldn't be known. The file created is a sparse file, which actually only contains the meta-data areas (boot sector, FATs, and root directory). The data portions won't be stored on the disk, but the file nevertheless will have the correct size. The resulting file can be copied later to a floppy disk or other device, or mounted through a loop device.
- D DRIVE-NUMBER**
Specify the BIOS drive number to be stored in the FAT boot sector. For hard disks and removable media it is usually 0x80–0xFF (0x80 is first hard disk C:, 0x81 is second hard disk D:, ...), for floppy devices or partitions to be used for floppy emulation it is 0x00–0x7F (0x00 is first floppy A:, 0x01 is second floppy B:).

-f NUMBER-OF-FATS

Specify the number of file allocation tables in the filesystem. The default is 2.

-F FAT-SIZE

Specifies the type of file allocation tables used (12, 16 or 32 bit). If nothing is specified, **mkfs.fat** will automatically select between 12, 16 and 32 bit, whatever fits better for the filesystem size.

-g HEADS/SECTORS-PER-TRACK

Specify *HEADS* and *SECTORS-PER-TRACK* numbers which represents disk geometry of *DEVICE*. Both numbers are stored into the FAT boot sector. Number *SECTORS-PER-TRACK* is used also for aligning the total count of FAT sectors. By default disk geometry is read from *DEVICE* itself. If it is not available then *LBA-Assist Translation* and translation table from the *SD Card Part 2 File System Specification* based on total number of disk sectors is used.

-h NUMBER-OF-HIDDEN-SECTORS

Specify the number of so-called *hidden sectors*, as stored in the FAT boot sector: this number represents the beginning sector of the partition containing the file system. Normally this is an offset (in sectors) relative to the start of the disk, although for MBR logical volumes contained in an extended partition of type 0x05 (a non-LBA extended partition), a quirk in the MS-DOS implementation of FAT requires it to be relative to the partition's immediate containing Extended Boot Record. Boot code and other software handling FAT volumes may also rely on this field being set up correctly; most modern FAT implementations will ignore it. By default, if the *DEVICE* is a partition block device, **mkfs.fat** uses the partition offset relative to disk start. Otherwise, **mkfs.fat** assumes zero. Use this option to override this behaviour.

-i VOLUME-ID

Sets the volume ID of the newly created filesystem; *VOLUME-ID* is a 32-bit hexadecimal number (for example, 2e24ec82). The default is a number which depends on the filesystem creation time.

-I

Ignore and disable safety checks. By default **mkfs.fat** refuses to create a filesystem on a device with partitions or virtual mapping. **mkfs.fat** will complain and tell you that it refuses to work. This is different when using MO disks. One doesn't always need partitions on MO disks. The filesystem can go directly to the whole disk. Under other OSes this is known as the *superfloppy* format. This switch will force **mkfs.fat** to work properly.

-l FILENAME

Read the bad blocks list from *FILENAME*.

-m MESSAGE-FILE

Sets the message the user receives on attempts to boot this filesystem without having properly installed an operating system. The message file must not exceed 418 bytes once line feeds have been converted to carriage return-line feed combinations, and tabs have been expanded. If the filename is a hyphen (-), the text is taken from standard input.

-M FAT-MEDIA-TYPE

Specify the media type to be stored in the FAT boot sector. This value is usually 0xF8 for hard disks and is 0xF0 or a value from 0xF9 to 0xFF for floppies or partitions to be used for floppy emulation.

--mbr[=y|yes|n|no|a|auto]

Fill (fake) MBR table with disk signature one partition which starts at sector 0 (includes MBR itself) and spans whole disk device. It is needed only for non-removable disks used on Microsoft Windows systems and only when formatting whole unpartitioned disk. Location of the disk signature and partition table overlaps with the end of the first FAT sector (boot code location), therefore there is no additional space usage. Default is *auto* mode in which **mkfs.fat** put MBR table only for non-removable disks when formatting whole unpartitioned disk.

-n VOLUME-NAME

Sets the volume name (label) of the filesystem. The volume name can be up to 11 characters long. Supplying an empty string, a string consisting only of white space or the string "NO NAME" as *VOLUME-NAME* has the same effect as not giving the **-n** option. The default is no label.

--codepage=PAGE

Use DOS codepage *PAGE* to encode label. By default codepage 850 is used.

-r ROOT-DIR-ENTRIES

Select the minimal number of entries available in the root directory. The default is 112 or 224 for floppies and 512 for hard disks. Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-R NUMBER-OF-RESERVED-SECTORS

Select the minimal number of reserved sectors. With FAT32 format at least 2 reserved sectors are needed, the default is 32. Otherwise the default is 1 (only the boot sector). Note that this is minimal number and it may be increased by **mkfs.fat** due to alignment of structures. See also **mkfs.fat** option **-a**.

-s SECTORS-PER-CLUSTER

Specify the number of disk sectors per cluster. Must be a power of 2, i.e. 1, 2, 4, 8, ... 128.

-S LOGICAL-SECTOR-SIZE

Specify the number of bytes per logical sector. Must be a power of 2 and greater than or equal to 512, i.e. 512, 1024, 2048, 4096, 8192, 16384, or 32768. Values larger than 4096 are not conforming to the FAT file system specification and may not work everywhere.

-v Verbose execution.**--offset SECTOR**

Write the filesystem at a specific sector into the device file. This is useful for creating a filesystem in a partitioned disk image without having to set up a loop device.

--variant TYPE

Create a filesystem of variant *TYPE*. Acceptable values are *standard* and *atari* (in any combination of upper/lower case). See above under DESCRIPTION for the differences.

--help

Display option summary and exit.

--invariant

Use constants for normally randomly generated or time based data such as volume ID and creation time. Multiple runs of **mkfs.fat** on the same device create identical results with this option. Its main purpose is testing **mkfs.fat**.

BUGS

mkfs.fat can not create boot-able filesystems. This isn't as easy as you might think at first glance for various reasons and has been discussed a lot already. **mkfs.fat** simply will not support it ;)

SEE ALSO

fatlabel(8), **fsck.fat(8)**

HOME PAGE

The home for the **dosfstools** project is its GitHub project page (<https://github.com/dosfstools/dosfstools>).

AUTHORS

dosfstools were written by Werner Almesberger <werner.almesberger@lrc.di.epfl.ch>, Roman Hodek <Roman.Hodek@informatik.uni-erlangen.de>, and others. Current maintainers are Andreas Bombe <aeb@debian.org> and Pali Rohár <pali.rohar@gmail.com>.

NAME

mkfs.xfs – construct an XFS filesystem

SYNOPSIS

```
mkfs.xfs [ -b block_size_options ] [ -m global_metadata_options ] [ -d data_section_options ] [ -f ] [ -i inode_options ] [ -l log_section_options ] [ -n naming_options ] [ -p protofile ] [ -q ] [ -r realtime_section_options ] [ -s sector_size_options ] [ -L label ] [ -N ] [ -K ] device
```

```
mkfs.xfs -V
```

DESCRIPTION

mkfs.xfs constructs an XFS filesystem by writing on a special file using the values found in the arguments of the command line. It is invoked automatically by **mkfs(8)** when it is given the **-t xfs** option.

In its simplest (and most commonly used form), the size of the filesystem is determined from the disk driver. As an example, to make a filesystem with an internal log on the first partition on the first SCSI disk, use:

```
mkfs.xfs /dev/sda1
```

The metadata log can be placed on another device to reduce the number of disk seeks. To create a filesystem on the first partition on the first SCSI disk with a 10MiB log located on the first partition on the second SCSI disk, use:

```
mkfs.xfs -l logdev=/dev/sdb1,size=10m /dev/sda1
```

Each of the *option* elements in the argument list above can be given as multiple comma-separated suboptions if multiple suboptions apply to the same option. Equivalently, each main option can be given multiple times with different suboptions. For example, **-l internal,size=10m** and **-l internal -l size=10m** are equivalent.

In the descriptions below, sizes are given in sectors, bytes, blocks, kilobytes, megabytes, gigabytes, etc. Sizes are treated as hexadecimal if prefixed by 0x or 0X, octal if prefixed by 0, or decimal otherwise. The following lists possible multiplication suffixes:

- s** – multiply by sector size (default = 512, see **-s** option below).
- b** – multiply by filesystem block size (default = 4K, see **-b** option below).
- k** – multiply by one kilobyte (1,024 bytes).
- m** – multiply by one megabyte (1,048,576 bytes).
- g** – multiply by one gigabyte (1,073,741,824 bytes).
- t** – multiply by one terabyte (1,099,511,627,776 bytes).
- p** – multiply by one petabyte (1,024 terabytes).
- e** – multiply by one exabyte (1,048,576 terabytes).

When specifying parameters in units of sectors or filesystem blocks, the **-s** option or the **-b** option may be used to specify the size of the sector or block. If the size of the block or sector is not specified, the default sizes (block: 4KiB, sector: 512B) will be used.

Many feature options allow an optional argument of 0 or 1, to explicitly disable or enable the functionality.

OPTIONS

Options may be specified either on the command line or in a configuration file. Not all command line options can be specified in configuration files; only the command line options followed by a **[section]** label can be used in a configuration file.

Options that can be used in configuration files are grouped into related sections containing multiple options. The command line options and configuration files use the same option sections and grouping. Configuration file section names are listed in the command line option sections below. Option names and values are the same for both command line and configuration file specification.

Options specified are the combined set of command line parameters and configuration file parameters. Duplicated options will result in a respecification error, regardless of the location they were specified at.

-c configuration_file_option

This option specifies the files that mkfs configuration will be obtained from. The valid *configuration_file_option* is:

options=name

The configuration options will be sourced from the file specified by the *name* option string. This option can be used either an absolute or relative path to the configuration file to be read.

-b block_size_options**Section Name: [block]**

This option specifies the fundamental block size of the filesystem. The valid *block_size_option* is:

size=value

The filesystem block size is specified with a *value* in bytes. The default value is 4096 bytes (4 KiB), the minimum is 512, and the maximum is 65536 (64 KiB).

Although **mkfs.xfs** will accept any of these values and create a valid filesystem, XFS on Linux can only mount filesystems with pagesize or smaller blocks.

-m global_metadata_options**Section Name: [metadata]**

These options specify metadata format options that either apply to the entire filesystem or aren't easily characterised by a specific functionality group. The valid *global_metadata_options* are:

bigtime=value

This option enables filesystems that can handle inode timestamps from December 1901 to July 2486, and quota timer expirations from January 1970 to July 2486. The value is either 0 to disable the feature, or 1 to enable large timestamps.

If this feature is not enabled, the filesystem can only handle timestamps from December 1901 to January 2038, and quota timers from January 1970 to February 2106.

By default, **mkfs.xfs** will not enable this feature. If the option **-m crc=0** is used, the large timestamp feature is not supported and is disabled.

crc=value

This is used to create a filesystem which maintains and checks CRC information in all metadata objects on disk. The value is either 0 to disable the feature, or 1 to enable the use of CRCs.

CRCs enable enhanced error detection due to hardware issues, whilst the format changes also improves crash recovery algorithms and the ability of various tools to validate and repair metadata corruptions when they are found. The CRC algorithm used is CRC32c, so the overhead is dependent on CPU architecture as some CPUs have hardware acceleration of this algorithm. Typically the overhead of calculating and checking the CRCs is not noticeable in normal operation.

By default, **mkfs.xfs** will enable metadata CRCs.

Formatting a filesystem without CRCs selects the V4 format, which is deprecated and will be removed from upstream in September 2030. Distributors may choose to withdraw support for the V4 format earlier than this date. Several other options, noted below, are only tunable on V4 formats, and will be removed along with the V4 format itself.

finobt=value

This option enables the use of a separate free inode btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create a free inode btree in each allocation group.

The free inode btree mirrors the existing allocated inode btree index which indexes both used and free inodes. The free inode btree does not index used inodes, allowing faster, more consistent inode allocation performance as filesystems age.

By default, **mkfs.xfs** will create free inode btrees for filesystems created with the (default) **-m crc=1** option set. When the option **-m crc=0** is used, the free inode btree feature is not supported and is disabled.

inobtcount=value

This option causes the filesystem to record the number of blocks used by the inode btree and the free inode btree. This can be used to reduce mount times when the free inode btree is enabled.

By default, **mkfs.xfs** will not enable this option. This feature is only available for filesystems created with the (default) **-m finobt=1** option set. When the option **-m finobt=0** is used, the inode btree counter feature is not supported and is disabled.

uuid=value

Use the given value as the filesystem UUID for the newly created filesystem. The default is to generate a random UUID.

rmapbt=value

This option enables the creation of a reverse-mapping btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create the btree.

The reverse mapping btree maps filesystem blocks to the owner of the filesystem block. Most of the mappings will be to an inode number and an offset, though there will also be mappings to filesystem metadata. This secondary metadata can be used to validate the primary metadata or to pinpoint exactly which data has been lost when a disk error occurs.

By default, **mkfs.xfs** will not create reverse mapping btrees. This feature is only available for filesystems created with the (default) **-m crc=1** option set. When the option **-m crc=0** is used, the reverse mapping btree feature is not supported and is disabled.

reflink=value

This option enables the use of a separate reference count btree index in each allocation group. The value is either 0 to disable the feature, or 1 to create a reference count btree in each allocation group.

The reference count btree enables the sharing of physical extents between the data forks of different files, which is commonly known as "reflink". Unlike traditional Unix filesystems which assume that every inode and logical block pair map to a unique physical block, a reflink-capable XFS filesystem removes the uniqueness requirement, allowing up to four billion arbitrary inode/logical block pairs to map to a physical block. If a program tries to write to a multiply-referenced block in a file, the write will be redirected to a new block, and that file's logical-to-physical mapping will be changed to the new block ("copy on write"). This feature enables the creation of per-file snapshots and deduplication. It is only available for the data forks of regular files.

By default, **mkfs.xfs** will create reference count btrees and therefore will enable the reflink feature. This feature is only available for filesystems created with the (default) **-m crc=1** option set. When the option **-m crc=0** is used, the reference count btree feature is not supported and reflink is disabled.

Note: the filesystem DAX mount option (**-o dax**) is incompatible with reflink-enabled XFS filesystems. To use filesystem DAX with XFS, specify the **-m reflink=0** option to **mkfs.xfs** to disable the reflink feature.

-d *data_section_options*

Section Name: *[data]*

These options specify the location, size, and other parameters of the data section of the filesystem. The valid *data_section_options* are:

agcount=value

This is used to specify the number of allocation groups. The data section of the filesystem is divided into allocation groups to improve the performance of XFS. More allocation groups imply that more parallelism can be achieved when allocating blocks and inodes. The minimum allocation group size is 16 MiB; the maximum size is just under 1 TiB. The data section of the filesystem is divided into *value* allocation groups (default value is scaled automatically based on the underlying device size).

agsize=value

This is an alternative to using the **agcount** suboption. The *value* is the desired size of the allocation group expressed in bytes (usually using the **m** or **g** suffixes). This value must be a multiple of the filesystem block size, and must be at least 16MiB, and no more than 1TiB, and may be automatically adjusted to properly align with the stripe geometry. The **agcount** and **agsize** suboptions are mutually exclusive.

cowextsize=value

Set the copy-on-write extent size hint on all inodes created by **mkfs.xfs**. The value must be provided in units of filesystem blocks. If the value is zero, the default value (currently 32 blocks) will be used. Directories will pass on this hint to newly created regular files and directories.

name=value

This can be used to specify the name of the special file containing the filesystem. In this case, the log section must be specified as **internal** (with a size, see the **-l** option below) and there can be no real-time section.

file[=value]

This is used to specify that the file given by the **name** suboption is a regular file. The *value* is either 0 or 1, with 1 signifying that the file is regular. This suboption is used only to make a filesystem image. If the *value* is omitted then 1 is assumed.

size=value

This is used to specify the size of the data section. This suboption is required if **-d file[=1]** is given. Otherwise, it is only needed if the filesystem should occupy less space than the size of the special file.

sunit=value

This is used to specify the stripe unit for a RAID device or a logical volume. The *value* has to be specified in 512-byte block units. Use the **su** suboption to specify the stripe unit size in bytes. This suboption ensures that data allocations will be stripe unit aligned when the current end of file is being extended and the file size is larger than 512KiB. Also inode allocations and the internal log will be stripe unit aligned.

su=value

This is an alternative to using **sunit**. The **su** suboption is used to specify the stripe unit for a RAID device or a striped logical volume. The *value* has to be specified in bytes, (usually using the **m** or **g** suffixes). This *value* must be a multiple of the filesystem block size.

swidth=value

This is used to specify the stripe width for a RAID device or a striped logical volume. The *value* has to be specified in 512-byte block units. Use the **sw** suboption to specify the stripe width size in bytes. This suboption is required if **-d sunit** has been specified and it has to be a multiple of the **-d sunit** suboption.

sw=value

suboption is an alternative to using **swidth**. The **sw** suboption is used to specify the stripe width for a RAID device or striped logical volume. The *value* is expressed as a multiplier of the stripe unit, usually the same as the number of stripe members in the logical volume configuration, or data disks in a RAID device.

When a filesystem is created on a logical volume device, **mkfs.xfs** will automatically query the logical volume for appropriate **sunit** and **swidth** values.

noalign

This option disables automatic geometry detection and creates the filesystem without stripe geometry alignment even if the underlying storage device provides this information.

rtinherit=value

If *value* is set to 1, all inodes created by **mkfs.xfs** will be created with the realtime flag set. The default is 0. Directories will pass on this flag to newly created regular files and directories.

projinherit=value

All inodes created by **mkfs.xfs** will be assigned the project quota id provided in *value*. Directories will pass on the project id to newly created regular files and directories.

extszinherit=value

All inodes created by **mkfs.xfs** will have this *value* extent size hint applied. The value must be provided in units of filesystem blocks. Directories will pass on this hint to newly created regular files and directories.

daxinherit=value

If *value* is set to 1, all inodes created by **mkfs.xfs** will be created with the DAX flag set. The default is 0. Directories will pass on this flag to newly created regular files and directories. By default, **mkfs.xfs** will not enable DAX mode.

- f** Force overwrite when an existing filesystem is detected on the device. By default, **mkfs.xfs** will not write to the device if it suspects that there is a filesystem or partition table on the device already.

-i inode_options**Section Name: [inode]**

This option specifies the inode size of the filesystem, and other inode allocation parameters. The XFS inode contains a fixed-size part and a variable-size part. The variable-size part, whose size is affected by this option, can contain: directory data, for small directories; attribute data, for small attribute sets; symbolic link data, for small symbolic links; the extent list for the file, for files with a small number of extents; and the root of a tree describing the location of extents for the file, for

files with a large number of extents.

The valid *inode_options* are:

size=value | perblock=value

The inode size is specified either as a *value* in bytes with **size=** or as the number fitting in a filesystem block with **perblock=**. The minimum (and default) *value* is 256 bytes without crc, 512 bytes with crc enabled. The maximum *value* is 2048 (2 KiB) subject to the restriction that the inode size cannot exceed one half of the filesystem block size.

XFS uses 64-bit inode numbers internally; however, the number of significant bits in an inode number is affected by filesystem geometry. In practice, filesystem size and inode size are the predominant factors. The Linux kernel (on 32 bit hardware platforms) and most applications cannot currently handle inode numbers greater than 32 significant bits, so if no inode size is given on the command line, **mkfs.xfs** will attempt to choose a size such that inode numbers will be < 32 bits. If an inode size is specified, or if a filesystem is sufficiently large, **mkfs.xfs** will warn if this will create inode numbers > 32 significant bits.

maxpct=value

This specifies the maximum percentage of space in the filesystem that can be allocated to inodes. The default *value* is 25% for filesystems under 1TB, 5% for filesystems under 50TB and 1% for filesystems over 50TB.

In the default inode allocation mode, inode blocks are chosen such that inode numbers will not exceed 32 bits, which restricts the inode blocks to the lower portion of the filesystem. The data block allocator will avoid these low blocks to accommodate the specified maxpct, so a high value may result in a filesystem with nothing but inodes in a significant portion of the lower blocks of the filesystem. (This restriction is not present when the filesystem is mounted with the *inode64* option on 64-bit platforms).

Setting the value to 0 means that essentially all of the filesystem can become inode blocks, subject to inode32 restrictions.

This value can be modified with *xfs_growfs(8)*.

align[=value]

This is used to specify that inode allocation is or is not aligned. The *value* is either 0 or 1, with 1 signifying that inodes are allocated aligned. If the *value* is omitted, 1 is assumed. The default is that inodes are aligned. Aligned inode access is normally more efficient than unaligned access; alignment must be established at the time the filesystem is created, since inodes are allocated at that time. This option can be used to turn off inode alignment when the filesystem needs to be mountable by a version of IRIX that does not have the inode alignment feature (any release of IRIX before 6.2, and IRIX 6.2 without XFS patches).

This option is only tunable on the deprecated V4 format.

attr=value

This is used to specify the version of extended attribute inline allocation policy to be used. By default, this is 2, which uses an efficient algorithm for managing the available inline inode space between attribute and extent data.

The previous version 1, which has fixed regions for attribute and extent data, is kept for backwards compatibility with kernels older than version 2.6.16.

This option is only tunable on the deprecated V4 format.

projid32bit[=value]

This is used to enable 32bit quota project identifiers. The *value* is either 0 or 1, with 1 signifying that 32bit projid are to be enabled. If the value is omitted, 1 is assumed. (This default changed in release version 3.2.0.)

This option is only tunable on the deprecated V4 format.

sparse[=value]

Enable sparse inode chunk allocation. The *value* is either 0 or 1, with 1 signifying that sparse allocation is enabled. If the value is omitted, 1 is assumed. Sparse inode allocation is disabled by default. This feature is only available for filesystems formatted with **-m crc=1**.

When enabled, sparse inode allocation allows the filesystem to allocate smaller than the standard 64-inode chunk when free space is severely limited. This feature is useful for filesystems that might fragment free space over time such that no free extents are large enough to accommodate a chunk of 64 inodes. Without this feature enabled, inode allocations can fail with out of space errors under severe fragmented free space conditions.

-l log_section_options

Section Name: [log]

These options specify the location, size, and other parameters of the log section of the filesystem. The valid *log_section_options* are:

agnum=value

If the log is internal, allocate it in this AG.

internal[=value]

This is used to specify that the log section is a piece of the data section instead of being another device or logical volume. The *value* is either 0 or 1, with 1 signifying that the log is internal. If the *value* is omitted, 1 is assumed.

logdev=device

This is used to specify that the log section should reside on the *device* separate from the data section. The **internal=1** and **logdev** options are mutually exclusive.

size=value

This is used to specify the size of the log section.

If the log is contained within the data section and **size** isn't specified, **mkfs.xfs** will try to select a suitable log size depending on the size of the filesystem. The actual logsize depends on the filesystem block size and the directory block size.

Otherwise, the **size** suboption is only needed if the log section of the filesystem should occupy less space than the size of the special file. The *value* is specified in bytes or blocks, with a **b** suffix meaning multiplication by the filesystem block size, as described above. The overriding minimum value for size is 512 blocks. With some combinations of filesystem block size, inode size, and directory block size, the minimum log size is larger than 512 blocks.

version=value

This specifies the version of the log. The current default is 2, which allows for larger log buffer sizes, as well as supporting stripe-aligned log writes (see the **sunit** and **su** options, below).

The previous version 1, which is limited to 32k log buffers and does not support stripe-aligned writes, is kept for backwards compatibility with very old

2.4 kernels.

This option is only tunable on the deprecated V4 format.

sunit=*value*

This specifies the alignment to be used for log writes. The *value* has to be specified in 512-byte block units. Use the **su** suboption to specify the log stripe unit size in bytes. Log writes will be aligned on this boundary, and rounded up to this boundary. This gives major improvements in performance on some configurations such as software RAID5 when the **sunit** is specified as the filesystem block size. The equivalent byte value must be a multiple of the filesystem block size. Version 2 logs are automatically selected if the log **sunit** suboption is specified.

The **su** suboption is an alternative to using **sunit**.

su=*value*

This is used to specify the log stripe. The *value* has to be specified in bytes, (usually using the **s** or **b** suffixes). This value must be a multiple of the filesystem block size. Version 2 logs are automatically selected if the log **su** suboption is specified.

lazy-count=*value*

This changes the method of logging various persistent counters in the superblock. Under metadata intensive workloads, these counters are updated and logged frequently enough that the superblock updates become a serialization point in the filesystem. The *value* can be either 0 or 1.

With **lazy-count=1**, the superblock is not modified or logged on every change of the persistent counters. Instead, enough information is kept in other parts of the filesystem to be able to maintain the persistent counter values without needed to keep them in the superblock. This gives significant improvements in performance on some configurations. The default *value* is 1 (on) so you must specify **lazy-count=0** if you want to disable this feature for older kernels which don't support it.

This option is only tunable on the deprecated V4 format.

-n *naming_options*

Section Name: *[naming]*

These options specify the version and size parameters for the naming (directory) area of the filesystem. The valid *naming_options* are:

size=*value*

The directory block size is specified with a *value* in bytes. The block size must be a power of 2 and cannot be less than the filesystem block size. The default size *value* for version 2 directories is 4096 bytes (4 KiB), unless the filesystem block size is larger than 4096, in which case the default *value* is the filesystem block size. For version 1 directories the block size is the same as the filesystem block size.

version=*value*

The naming (directory) version *value* can be either 2 or 'ci', defaulting to 2 if unspecified. With version 2 directories, the directory block size can be any power of 2 size from the filesystem block size up to 65536.

The **version=ci** option enables ASCII only case-insensitive filename lookup and version 2 directories. Filenames are case-preserving, that is, the names are stored in directories using the case they were created with.

Note: Version 1 directories are not supported.

f*type*=*value*

This feature allows the inode type to be stored in the directory structure so that the **readdir**(3) and **getdents**(2) do not need to look up the inode to determine the inode type.

The *value* is either 0 or 1, with 1 signifying that filetype information will be stored in the directory structure. The default value is 1.

When CRCs are enabled (the default), the ftype functionality is always enabled, and cannot be turned off.

In other words, this option is only tunable on the deprecated V4 format.

-p *protofile*

If the optional **-p** *protofile* argument is given, **mkfs.xfs** uses *protofile* as a prototype file and takes its directions from that file. The blocks and inodes specifiers in the *protofile* are provided for backwards compatibility, but are otherwise unused. The syntax of the protofile is defined by a number of tokens separated by spaces or newlines. Note that the line numbers are not part of the syntax but are meant to help you in the following discussion of the file contents.

```

1      /stand/diskboot
2      4872 110
3      d--777 3 1
4      usr      d--777 3 1
5      sh       ---755 3 1 /bin/sh
6      ken      d--755 6 1
7      $
8      b0      b--644 3 1 0 0
9      c0      c--644 3 1 0 0
10     fifo     p--644 3 1
11     slink    l--644 3 1 /a/symbolic/link
12     : This is a comment line
13     $
14     $
```

Line 1 is a dummy string. (It was formerly the bootfilename.) It is present for backward compatibility; boot blocks are not used on SGI systems.

Note that some string of characters must be present as the first line of the proto file to cause it to be parsed correctly; the value of this string is immaterial since it is ignored.

Line 2 contains two numeric values (formerly the numbers of blocks and inodes). These are also merely for backward compatibility: two numeric values must appear at this point for the proto file to be correctly parsed, but their values are immaterial since they are ignored.

The lines 3 through 11 specify the files and directories you want to include in this filesystem. Line 3 defines the root directory. Other directories and files that you want in the filesystem are indicated by lines 4 through 6 and lines 8 through 10. Line 11 contains symbolic link syntax.

Notice the dollar sign (\$) syntax on line 7. This syntax directs the **mkfs.xfs** command to terminate the branch of the filesystem it is currently on and then continue from the directory specified by the next line, in this case line 8. It must be the last character on a line. The colon on line 12 introduces a comment; all characters up until the following newline are ignored. Note that this means you cannot have a file in a prototype file whose name contains a colon. The \$ on lines 13 and 14 end the process, since no additional specifications follow.

File specifications provide the following:

- * file mode
- * user ID

* group ID
 * the file's beginning contents

A 6-character string defines the mode for a file. The first character of this string defines the file type. The character range for this first character is **-bcdpl**. A file may be a regular file, a block special file, a character special file, directory files, named pipes (first-in, first out files), and symbolic links. The second character of the mode string is used to specify setuserID mode, in which case it is **u**. If setuserID mode is not specified, the second character is **-**. The third character of the mode string is used to specify the setgroupID mode, in which case it is **g**. If setgroupID mode is not specified, the third character is **-**. The remaining characters of the mode string are a three digit octal number. This octal number defines the owner, group, and other read, write, and execute permissions for the file, respectively. For more information on file permissions, see the **chmod(1)** command.

Following the mode character string are two decimal number tokens that specify the user and group IDs of the file's owner.

In a regular file, the next token specifies the pathname from which the contents and size of the file are copied. In a block or character special file, the next token are two decimal numbers that specify the major and minor device numbers. When a file is a symbolic link, the next token specifies the contents of the link.

When the file is a directory, the **mkfs.xfs** command creates the entries **dot** (.) and **dot-dot** (..) and then reads the list of names and file specifications in a recursive manner for all of the entries in the directory. A scan of the protofile is always terminated with the dollar (\$) token.

-q Quiet option. Normally **mkfs.xfs** prints the parameters of the filesystem to be constructed; the **-q** flag suppresses this.

-r realtime_section_options

Section Name: *[realtime]*

These options specify the location, size, and other parameters of the real-time section of the filesystem. The valid *realtime_section_options* are:

rtdev=device

This is used to specify the *device* which should contain the real-time section of the filesystem. The suboption value is the name of a block device.

extsize=value

This is used to specify the size of the blocks in the real-time section of the filesystem. This *value* must be a multiple of the filesystem block size. The minimum allowed size is the filesystem block size or 4 KiB (whichever is larger); the default size is the stripe width for striped volumes or 64 KiB for non-striped volumes; the maximum allowed size is 1 GiB. The real-time extent size should be carefully chosen to match the parameters of the physical media used.

size=value

This is used to specify the size of the real-time section. This suboption is only needed if the real-time section of the filesystem should occupy less space than the size of the partition or logical volume containing the section.

noalign

This option disables stripe size detection, enforcing a realtime device with no stripe geometry.

-s sector_size_options

Section Name: *[sector]*

This option specifies the fundamental sector size of the filesystem. The valid *sector_size_option* is:

size=value

The sector size is specified with a *value* in bytes. The default *sector_size* is 512 bytes. The minimum value for sector size is 512; the maximum is 32768 (32 KiB). The *sector_size* must be a power of 2 size and cannot be made larger than the filesystem block size.

-L label

Set the filesystem *label*. XFS filesystem labels can be at most 12 characters long; if *label* is longer than 12 characters, **mkfs.xfs** will not proceed with creating the filesystem. Refer to the **mount(8)** and **xfs_admin(8)** manual entries for additional information.

- N** Causes the file system parameters to be printed out without really creating the file system.
- K** Do not attempt to discard blocks at mkfs time.
- V** Prints the version number and exits.

Configuration File Format

The configuration file uses a basic INI format to specify sections and options within a section. Section and option names are case sensitive. Section names must not contain whitespace. Options are name-value pairs, ended by the first whitespace in the line. Option names cannot contain whitespace. Full line comments can be added by starting a line with a # symbol. If values contain whitespace, then it must be quoted.

The following example configuration file sets the block size to 4096 bytes, turns on reverse mapping btrees and sets the inode size to 2048 bytes.

```
# Example mkfs.xfs configuration file
```

```
[block]
size=4k
```

```
[metadata]
rmapbt=1
```

```
[inode]
size=2048
```

SEE ALSO

xfs(5), mkfs(8), mount(8), xfs_info(8), xfs_admin(8).

BUGS

With a prototype file, it is not possible to specify hard links.

NAME

modinfo – Show information about a Linux Kernel module

SYNOPSIS

modinfo [**-0**] [**-F field**] [**-k kernel**] [modulename|filename...]

modinfo -V

modinfo -h

DESCRIPTION

modinfo extracts information from the Linux Kernel modules given on the command line. If the module name is not a filename, then the `/lib/modules/version` directory is searched, as is also done by **modprobe(8)** when loading kernel modules.

modinfo by default lists each attribute of the module in form *fieldname* : *value*, for easy reading. The filename is listed the same way (although it's not really an attribute).

This version of **modinfo** can understand modules of any Linux Kernel architecture.

OPTIONS

-V, --version

Print the modinfo version.

-F, --field

Only print this field value, one per line. This is most useful for scripts. Field names are case-insensitive. Common fields (which may not be in every module) include author, description, license, parm, depends, and alias. There are often multiple parm, alias and depends fields. The special field filename lists the filename of the module.

-b basedir, --basedir basedir

Root directory for modules, / by default.

-k kernel

Provide information about a kernel other than the running one. This is particularly useful for distributions needing to extract information from a newly installed (but not yet running) set of kernel modules. For example, you wish to find which firmware files are needed by various modules in a new kernel for which you must make an initrd/initramfs image prior to booting.

-0, --null

Use the ASCII zero character to separate field values, instead of a new line. This is useful for scripts, since a new line can theoretically appear inside a field.

-a --author, -d --description, -l --license, -p --parameters, -n --filename

These are shortcuts for the **--field** flag's author, description, license, parm and filename arguments, to ease the transition from the old modutils **modinfo**.

COPYRIGHT

This manual page originally Copyright 2003, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

modprobe(8)

AUTHORS

Jon Masters <jcm@jonmasters.org>

Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>

Developer

NAME

modprobe – Add and remove modules from the Linux Kernel

SYNOPSIS

```
modprobe [-v] [-V] [-C config-file] [-n] [-i] [-q] [-b] [modulename] [module parameters...]
modprobe [-r] [-v] [-n] [-i] [modulename...]
modprobe [-c]
modprobe [--dump-modversions] [filename]
```

DESCRIPTION

modprobe intelligently adds or removes a module from the Linux kernel: note that for convenience, there is no difference between _ and – in module names (automatic underscore conversion is performed). **modprobe** looks in the module directory /lib/modules/`uname -r` for all the modules and other files, except for the optional configuration files in the /etc/modprobe.d directory (see **modprobe.d(5)**). **modprobe** will also use module options specified on the kernel command line in the form of <module>,<option> and blacklists in the form of modprobe.blacklist=<module>.

Note that unlike in 2.4 series Linux kernels (which are not supported by this tool) this version of **modprobe** does not do anything to the module itself: the work of resolving symbols and understanding parameters is done inside the kernel. So module failure is sometimes accompanied by a kernel message: see **dmesg(8)**.

modprobe expects an up-to-date modules.dep.bin file as generated by the corresponding **depmod** utility shipped along with **modprobe** (see **depmod(8)**). This file lists what other modules each module needs (if any), and **modprobe** uses this to add or remove these dependencies automatically.

If any arguments are given after the *modulename*, they are passed to the kernel (in addition to any options listed in the configuration file).

OPTIONS**-a, --all**

Insert all module names on the command line.

-b, --use-blacklist

This option causes **modprobe** to apply the **blacklist** commands in the configuration files (if any) to module names as well. It is usually used by **udev(7)**.

-C, --config

This option overrides the default configuration directory (/etc/modprobe.d).

This option is passed through **install** or **remove** commands to other **modprobe** commands in the MODPROBE_OPTIONS environment variable.

-c, --showconfig

Dump out the effective configuration from the config directory and exit.

--dump-modversions

Print out a list of module versioning information required by a module. This option is commonly used by distributions in order to package up a Linux kernel module using module versioning deps.

-d, --dirname

Root directory for modules, / by default.

--first-time

Normally, **modprobe** will succeed (and do nothing) if told to insert a module which is already present or to remove a module which isn't present. This is ideal for simple scripts; however, more complicated scripts often want to know whether **modprobe** really did something: this option makes modprobe fail in the case that it actually didn't do anything.

--force-vermagic

Every module contains a small string containing important information, such as the kernel and compiler versions. If a module fails to load and the kernel complains that the "version magic" doesn't

match, you can use this option to remove it. Naturally, this check is there for your protection, so using this option is dangerous unless you know what you're doing.

This applies to any modules inserted: both the module (or alias) on the command line and any modules on which it depends.

--force-modversion

When modules are compiled with CONFIG_MODVERSIONS set, a section detailing the versions of every interface used by (or supplied by) the module is created. If a module fails to load and the kernel complains that the module disagrees about a version of some interface, you can use "--force-modversion" to remove the version information altogether. Naturally, this check is there for your protection, so using this option is dangerous unless you know what you're doing.

This applies any modules inserted: both the module (or alias) on the command line and any modules on which it depends.

-f, --force

Try to strip any versioning information from the module which might otherwise stop it from loading: this is the same as using both **--force-vermagic** and **--force-modversion**. Naturally, these checks are there for your protection, so using this option is dangerous unless you know what you are doing.

This applies to any modules inserted: both the module (or alias) on the command line and any modules it on which it depends.

-i, --ignore-install, --ignore-remove

This option causes **modprobe** to ignore **install** and **remove** commands in the configuration file (if any) for the module specified on the command line (any dependent modules are still subject to commands set for them in the configuration file). Both **install** and **remove** commands will currently be ignored when this option is used regardless of whether the request was more specifically made with only one or other (and not both) of **--ignore-install** or **--ignore-remove**. See **modprobe.d(5)**.

-n, --dry-run, --show

This option does everything but actually insert or delete the modules (or run the install or remove commands). Combined with **-v**, it is useful for debugging problems. For historical reasons both **--dry-run** and **--show** actually mean the same thing and are interchangeable.

-q, --quiet

With this flag, **modprobe** won't print an error message if you try to remove or insert a module it can't find (and isn't an alias or **install/remove** command). However, it will still return with a non-zero exit status. The kernel uses this to opportunistically probe for modules which might exist using **request_module**.

-R, --resolve-alias

Print all module names matching an alias. This can be useful for debugging module alias problems.

-r, --remove

This option causes **modprobe** to remove rather than insert a module. If the modules it depends on are also unused, **modprobe** will try to remove them too. Unlike insertion, more than one module can be specified on the command line (it does not make sense to specify module parameters when removing modules).

There is usually no reason to remove modules, but some buggy modules require it. Your distribution kernel may not have been built to support removal of modules at all.

-S, --set-version

Set the kernel version, rather than using **uname(2)** to decide on the kernel version (which dictates where to find the modules).

--show-dependents

List the dependencies of a module (or alias), including the module itself. This produces a (possibly

empty) set of module filenames, one per line, each starting with "insmod" and is typically used by distributions to determine which modules to include when generating initrd/initramfs images. **Install** commands which apply are shown prefixed by "install". It does not run any of the install commands. Note that **modinfo(8)** can be used to extract dependencies of a module from the module itself, but knows nothing of aliases or install commands.

-s, --syslog

This option causes any error messages to go through the syslog mechanism (as LOG_DAEMON with level LOG_NOTICE) rather than to standard error. This is also automatically enabled when stderr is unavailable.

This option is passed through **install** or **remove** commands to other **modprobe** commands in the MODPROBE_OPTIONS environment variable.

-V, --version

Show version of program and exit.

-v, --verbose

Print messages about what the program is doing. Usually **modprobe** only prints messages if something goes wrong.

This option is passed through **install** or **remove** commands to other **modprobe** commands in the MODPROBE_OPTIONS environment variable.

ENVIRONMENT

The MODPROBE_OPTIONS environment variable can also be used to pass arguments to **modprobe**.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

modprobe.d(5), **insmod(8)**, **rmmmod(8)**, **lsmod(8)**, **modinfo(8)** **depmod(8)**

AUTHORS

Jon Masters <jcm@jonmasters.org>
Developer

Robby Workman <rworkman@slackware.com>
Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>
Developer

NAME

modprobe.d – Configuration directory for modprobe

SYNOPSIS

```
/lib/modprobe.d/*.conf
/usr/local/lib/modprobe.d/*.conf
/run/modprobe.d/*.conf
/etc/modprobe.d/*.conf
```

DESCRIPTION

Because the **modprobe** command can add or remove more than one module, due to modules having dependencies, we need a method of specifying what options are to be used with those modules. All files underneath the /etc/modprobe.d directory which end with the .conf extension specify those options as required. They can also be used to create convenient aliases: alternate names for a module, or they can override the normal **modprobe** behavior altogether for those with special requirements (such as inserting more than one module).

Note that module and alias names (like other module names) can have – or _ in them: both are interchangeable throughout all the module commands as underscore conversion happens automatically.

The format of files under modprobe.d is simple: one command per line, with blank lines and lines starting with #' ignored (useful for adding comments). A \' at the end of a line causes it to continue on the next line, which makes the file a bit neater.

COMMANDS*alias wildcard modulename*

This allows you to give alternate names for a module. For example: "alias my-mod really_long_modulename" means you can use "modprobe my-mod" instead of "modprobe really_long_modulename". You can also use shell-style wildcards, so "alias my-mod* really_long_modulename" means that "modprobe my-mod–something" has the same effect. You can't have aliases to other aliases (that way lies madness), but aliases can have options, which will be added to any other options.

Note that modules can also contain their own aliases, which you can see using **modinfo**. These aliases are used as a last resort (ie. if there is no real module, **install**, **remove**, or **alias** command in the configuration).

blacklist modulename

Modules can contain their own aliases: usually these are aliases describing the devices they support, such as "pci:123...". These "internal" aliases can be overridden by normal "alias" keywords, but there are cases where two or more modules both support the same devices, or a module invalidly claims to support a device that it does not: the **blacklist** keyword indicates that all of that particular module's internal aliases are to be ignored.

install modulename command...

This command instructs **modprobe** to run your command instead of inserting the module in the kernel as normal. The command can be any shell command: this allows you to do any kind of complex processing you might wish. For example, if the module "fred" works better with the module "barney" already installed (but it doesn't depend on it, so **modprobe** won't automatically load it), you could say "install fred /sbin/modprobe barney; /sbin/modprobe --ignore-install fred", which would do what you wanted. Note the **--ignore-install**, which stops the second **modprobe** from running the same **install** command again. See also **remove** below.

The long term future of this command as a solution to the problem of providing additional module dependencies is not assured and it is intended to replace this command with a warning about its eventual removal or deprecation at some point in a future release. Its use complicates the automated determination of module dependencies by distribution utilities, such as mkinitrd (because these now

need to somehow interpret what the **install** commands might be doing. In a perfect world, modules would provide all dependency information without the use of this command and work is underway to implement soft dependency support within the Linux kernel.

If you use the string "\$CMDLINE_OPTS" in the command, it will be replaced by any options specified on the modprobe command line. This can be useful because users expect "modprobe fred opt=1" to pass the "opt=1" arg to the module, even if there's an install command in the configuration file. So our above example becomes "install fred /sbin/modprobe barney; /sbin/modprobe ---ignore--install fred \$CMDLINE_OPTS"

options modulename option...

This command allows you to add options to the module *modulename* (which might be an alias) every time it is inserted into the kernel: whether directly (using **modprobe modulename**) or because the module being inserted depends on this module.

All options are added together: they can come from an **option** for the module itself, for an alias, and on the command line.

remove modulename command...

This is similar to the **install** command above, except it is invoked when "modprobe -r" is run.

softdep modulename pre: modules... post: modules...

The **softdep** command allows you to specify soft, or optional, module dependencies. *modulename* can be used without these optional modules installed, but usually with some features missing. For example, a driver for a storage HBA might require another module be loaded in order to use management features.

pre-deps and post-deps modules are lists of names and/or aliases of other modules that modprobe will attempt to install (or remove) in order before and after the main module given in the *modulename* argument.

Example: Assume "softdep c pre: a b post: d e" is provided in the configuration. Running "modprobe c" is now equivalent to "modprobe a b c d e" without the softdep. Flags such as --use-blacklist are applied to all the specified modules, while module parameters only apply to module c.

Note: if there are **install** or **remove** commands with the same *modulename* argument, **softdep** takes precedence.

COMPATIBILITY

A future version of kmod will come with a strong warning to avoid use of the **install** as explained above. This will happen once support for soft dependencies in the kernel is complete. That support will complement the existing softdep support within this utility by providing such dependencies directly within the modules.

COPYRIGHT

This manual page originally Copyright 2004, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

modprobe(8), **modules.dep(5)**

AUTHORS

Jon Masters <jcm@jonmasters.org>
Developer

Robby Workman <rworkman@slackware.com>
Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>
Developer

MODPROBE.D(5)

modprobe.d

MODPROBE.D(5)

NAME

/etc/modules - kernel modules to load at boot time

DESCRIPTION

The **/etc/modules** file contains the names of kernel modules that are to be loaded at boot time, one per line. Options can only be specified using **modprobe.d** configuration files. Lines beginning with a '#' are ignored.

EXAMPLE

```
# /etc/modules: kernel modules to load at boot time.  
#  
# This file contains the names of kernel modules that  
# should be loaded at boot time, one per line. Lines  
# beginning with "#" are ignored.
```

w83781d

```
3c509  
nf_nat_ftp
```

SEE ALSO

depmod(8) **modprobe(8)** **modprobe.d(5)**

NAME

more – file perusal filter for crt viewing

SYNOPSIS

more [options] *file* ...

DESCRIPTION

more is a filter for paging through text one screenful at a time. This version is especially primitive. Users should realize that **less(1)** provides **more(1)** emulation plus extensive enhancements.

OPTIONS

Options are also taken from the environment variable **MORE** (make sure to precede them with a dash (-)) but command-line options will override those.

-d, --silent

Prompt with "[Press space to continue, 'q' to quit.]", and display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.

-l, --logical

Do not pause after any line containing a ^L (form feed).

-e, --exit-on-eof

Exit on End-Of-File, enabled by default if not executed on terminal.

-f, --no-pause

Count logical lines, rather than screen lines (i.e., long lines are not folded).

-p, --print-over

Do not scroll. Instead, clear the whole screen and then display the text. Notice that this option is switched on automatically if the executable is named **page**.

-c, --clean-print

Do not scroll. Instead, paint each screen from the top, clearing the remainder of each line as it is displayed.

-s, --squeeze

Squeeze multiple blank lines into one.

-u, --plain

Suppress underlining. This option is silently ignored as backwards compatibility.

-n, --lines *number*

Specify the *number* of lines per screenful. The *number* argument is a positive decimal integer. The **--lines** option shall override any values obtained from any other source, such as number of lines reported by terminal.

number

A numeric option means the same as **--lines** option argument.

+*number*

Start displaying each file at line *number*.

+/*string*

The *string* to be searched in each file before starting to display it.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

COMMANDS

Interactive commands for **more** are based on **vi(1)**. Some commands may be preceded by a decimal number, called k in the descriptions below. In the following descriptions, **^X** means **control-X**.

h or ?

Help; display a summary of these commands. If you forget all other commands, remember this one.

SPACE

Display next k lines of text. Defaults to current screen size.

z

Display next k lines of text. Defaults to current screen size. Argument becomes new default.

RETURN

Display next k lines of text. Defaults to 1. Argument becomes new default.

d or ^D

Scroll k lines. Default is current scroll size, initially 11. Argument becomes new default.

q or Q or INTERRUPT

Exit.

s

Skip forward k lines of text. Defaults to 1.

f

Skip forward k screenfuls of text. Defaults to 1.

b or ^B

Skip backwards k screenfuls of text. Defaults to 1. Only works with files, not pipes.

,

Go to the place where the last search started.

=

Display current line number.

/pattern

Search for kth occurrence of regular expression. Defaults to 1.

n

Search for kth occurrence of last regular expression. Defaults to 1.

!command or :!command

Execute *command* in a subshell.

v

Start up an editor at current line. The editor is taken from the environment variable **VISUAL** if defined, or **EDITOR** if **VISUAL** is not defined, or defaults to **vi(1)** if neither **VISUAL** nor **EDITOR** is defined.

^L

Redraw screen.

:n

Go to kth next file. Defaults to 1.

:p

Go to kth previous file. Defaults to 1.

:f

Display current file name and line number.

.

Repeat previous command.

ENVIRONMENT

The **more** command respects the following environment variables, if they exist:

MORE

This variable may be set with favored options to **more**.

SHELL

Current shell in use (normally set by the shell at login time).

TERM

The terminal type used by **more** to get the terminal characteristics necessary to manipulate the screen.

VISUAL

The editor the user prefers. Invoked when command key *v* is pressed.

EDITOR

The editor of choice when **VISUAL** is not specified.

HISTORY

The **more** command appeared in 3.0BSD. This man page documents **more** version 5.19 (Berkeley 6/29/88), which is currently in use in the Linux community. Documentation was produced using several other versions of the man page, and extensive inspection of the source code.

AUTHORS

Eric Shienbrood, UC Berkeley.

Modified by Geoff Peck, UCB to add underlining, single spacing.

Modified by John Foderaro, UCB to add -c and MORE environment variable.

SEE ALSO

[less\(1\)](#), [vi\(1\)](#)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **more** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

`mount` – mount filesystem

LIBRARY

Standard C library (`libc, -lc`)

SYNOPSIS

```
#include <sys/mount.h>
int mount(const char *source, const char *target,
          const char *filesystemtype, unsigned long mountflags,
          const void *_Nullable data);
```

DESCRIPTION

`mount()` attaches the filesystem specified by *source* (which is often a pathname referring to a device, but can also be the pathname of a directory or file, or a dummy string) to the location (a directory or file) specified by the pathname in *target*.

Appropriate privilege (Linux: the **CAP_SYS_ADMIN** capability) is required to mount filesystems.

Values for the *filesystemtype* argument supported by the kernel are listed in */proc/filesystems* (e.g., "btrfs", "ext4", "jfs", "xfs", "vfat", "fuse", "tmpfs", "cgroup", "proc", "mqueue", "nfs", "cifs", "iso9660"). Further types may become available when the appropriate modules are loaded.

The *data* argument is interpreted by the different filesystems. Typically it is a string of comma-separated options understood by this filesystem. See **mount(8)** for details of the options available for each filesystem type. This argument may be specified as NULL, if there are no options.

A call to `mount()` performs one of a number of general types of operation, depending on the bits specified in *mountflags*. The choice of which operation to perform is determined by testing the bits set in *mountflags*, with the tests being conducted in the order listed here:

- Remount an existing mount: *mountflags* includes **MS_REMOUNT**.
- Create a bind mount: *mountflags* includes **MS_BIND**.
- Change the propagation type of an existing mount: *mountflags* includes one of **MS_SHARED**, **MS_PRIVATE**, **MS_SLAVE**, or **MS_UNBINDABLE**.
- Move an existing mount to a new location: *mountflags* includes **MS_MOVE**.
- Create a new mount: *mountflags* includes none of the above flags.

Each of these operations is detailed later in this page. Further flags may be specified in *mountflags* to modify the behavior of `mount()`, as described below.

Additional mount flags

The list below describes the additional flags that can be specified in *mountflags*. Note that some operation types ignore some or all of these flags, as described later in this page.

MS_DIRSYNC (since Linux 2.5.19)

Make directory changes on this filesystem synchronous. (This property can be obtained for individual directories or subtrees using **chattr(1)**.)

MS_LAZYTIME (since Linux 4.0)

Reduce on-disk updates of inode timestamps (atime, mtime, ctime) by maintaining these changes only in memory. The on-disk timestamps are updated only when:

- the inode needs to be updated for some change unrelated to file timestamps;
- the application employs **fsync(2)**, **syncfs(2)**, or **sync(2)**;
- an undeleted inode is evicted from memory; or
- more than 24 hours have passed since the inode was written to disk.

This mount option significantly reduces writes needed to update the inode's timestamps, especially mtime and atime. However, in the event of a system crash, the atime and mtime fields on disk

might be out of date by up to 24 hours.

Examples of workloads where this option could be of significant benefit include frequent random writes to preallocated files, as well as cases where the **MS_STRICTATIME** mount option is also enabled. (The advantage of combining **MS_STRICTATIME** and **MS_LAZYTIME** is that **stat(2)** will return the correctly updated atime, but the atime updates will be flushed to disk only in the cases listed above.)

MS_MANDLOCK

Permit mandatory locking on files in this filesystem. (Mandatory locking must still be enabled on a per-file basis, as described in **fctl(2)**.) Since Linux 4.5, this mount option requires the **CAP_SYS_ADMIN** capability and a kernel configured with the **CONFIG_MANDATORY_FILE_LOCKING** option. Mandatory locking has been fully deprecated in Linux 5.15, so this flag should be considered deprecated.

MS_NOATIME

Do not update access times for (all types of) files on this filesystem.

MS_NODEV

Do not allow access to devices (special files) on this filesystem.

MS_NODIRATIME

Do not update access times for directories on this filesystem. This flag provides a subset of the functionality provided by **MS_NOATIME**; that is, **MS_NOATIME** implies **MS_NODIRATIME**.

MS_NOEXEC

Do not allow programs to be executed from this filesystem.

MS_NOSUID

Do not honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem. In addition, SELinux domain transitions require the permission *nosuid_transition*, which in turn needs also the policy capability *nnp_nosuid_transition*.

MS_RDONLY

Mount filesystem read-only.

MS_REC (since Linux 2.4.11)

Used in conjunction with **MS_BIND** to create a recursive bind mount, and in conjunction with the propagation type flags to recursively change the propagation type of all of the mounts in a subtree. See below for further details.

MS_RELATIME (since Linux 2.6.20)

When a file on this filesystem is accessed, update the file's last access time (atime) only if the current value of atime is less than or equal to the file's last modification time (mtime) or last status change time (ctime). This option is useful for programs, such as **mutt(1)**, that need to know when a file has been read since it was last modified. Since Linux 2.6.30, the kernel defaults to the behavior provided by this flag (unless **MS_NOATIME** was specified), and the **MS_STRICTATIME** flag is required to obtain traditional semantics. In addition, since Linux 2.6.30, the file's last access time is always updated if it is more than 1 day old.

MS_SILENT (since Linux 2.6.17)

Suppress the display of certain (*printk()*) warning messages in the kernel log. This flag supersedes the misnamed and obsolete **MS_VERBOSE** flag (available since Linux 2.4.12), which has the same meaning.

MS_STRICTATIME (since Linux 2.6.30)

Always update the last access time (atime) when files on this filesystem are accessed. (This was the default behavior before Linux 2.6.30.) Specifying this flag overrides the effect of setting the **MS_NOATIME** and **MS_RELATIME** flags.

MS_SYNCHRONOUS

Make writes on this filesystem synchronous (as though the **O_SYNC** flag to **open(2)** was specified for all file opens to this filesystem).

MS_NOSYMFOLLOW (since Linux 5.10)

Do not follow symbolic links when resolving paths. Symbolic links can still be created, and **readlink(1)**, **readlink(2)**, **realpath(1)**, and **realpath(3)** all still work properly.

From Linux 2.4 onward, some of the above flags are settable on a per-mount basis, while others apply to the superblock of the mounted filesystem, meaning that all mounts of the same filesystem share those flags. (Previously, all of the flags were per-superblock.)

The per-mount-point flags are as follows:

- Since Linux 2.4: **MS_NODEV**, **MS_NOEXEC**, and **MS_NOSUID** flags are settable on a per-mount-point basis.
- Additionally, since Linux 2.6.16: **MS_NOATIME** and **MS_NODIRATIME**.
- Additionally, since Linux 2.6.20: **MS_RELATIME**.

The following flags are per-superblock: **MS_DIRSYNC**, **MS_LAZYTIME**, **MS_MANDLOCK**, **MS_SILENT**, and **MS_SYNCHRONOUS**. The initial settings of these flags are determined on the first mount of the filesystem, and will be shared by all subsequent mounts of the same filesystem. Subsequently, the settings of the flags can be changed via a remount operation (see below). Such changes will be visible via all mounts associated with the filesystem.

Since Linux 2.6.16, **MS_RDONLY** can be set or cleared on a per-mount-point basis as well as on the underlying filesystem superblock. The mounted filesystem will be writable only if neither the filesystem nor the mountpoint are flagged as read-only.

Remounting an existing mount

An existing mount may be remounted by specifying **MS_REMOUNT** in *mountflags*. This allows you to change the *mountflags* and *data* of an existing mount without having to unmount and remount the filesystem. *target* should be the same value specified in the initial **mount()** call.

The *source* and *filesystemtype* arguments are ignored.

The *mountflags* and *data* arguments should match the values used in the original **mount()** call, except for those parameters that are being deliberately changed.

The following *mountflags* can be changed: **MS_LAZYTIME**, **MS_MANDLOCK**, **MS_NOATIME**, **MS_NODEV**, **MS_NODIRATIME**, **MS_NOEXEC**, **MS_NOSUID**, **MS_RELATIME**, **MS_RDONLY**, **MS_STRICTATIME** (whose effect is to clear the **MS_NOATIME** and **MS_RELATIME** flags), and **MS_SYNCHRONOUS**. Attempts to change the setting of the **MS_DIRSYNC** and **MS_SILENT** flags during a remount are silently ignored. Note that changes to per-superblock flags are visible via all mounts of the associated filesystem (because the per-superblock flags are shared by all mounts).

Since Linux 3.17, if none of **MS_NOATIME**, **MS_NODIRATIME**, **MS_RELATIME**, or **MS_STRICTATIME** is specified in *mountflags*, then the remount operation preserves the existing values of these flags (rather than defaulting to **MS_RELATIME**).

Since Linux 2.6.26, the **MS_REMOUNT** flag can be used with **MS_BIND** to modify only the per-mount-point flags. This is particularly useful for setting or clearing the "read-only" flag on a mount without changing the underlying filesystem. Specifying *mountflags* as:

```
MS_REMOUNT | MS_BIND | MS_RDONLY
```

will make access through this mountpoint read-only, without affecting other mounts.

Creating a bind mount

If *mountflags* includes **MS_BIND** (available since Linux 2.4), then perform a bind mount. A bind mount makes a file or a directory subtree visible at another point within the single directory hierarchy. Bind mounts may cross filesystem boundaries and span **chroot(2)** jails.

The *filesystemtype* and *data* arguments are ignored.

The remaining bits (other than **MS_REC**, described below) in the *mountflags* argument are also ignored. (The bind mount has the same mount options as the underlying mount.) However, see the discussion of re-mounting above, for a method of making an existing bind mount read-only.

By default, when a directory is bind mounted, only that directory is mounted; if there are any submounts under the directory tree, they are not bind mounted. If the **MS_REC** flag is also specified, then a recursive bind mount operation is performed: all submounts under the *source* subtree (other than unbindable mounts) are also bind mounted at the corresponding location in the *target* subtree.

Changing the propagation type of an existing mount

If *mountflags* includes one of **MS_SHARED**, **MS_PRIVATE**, **MS_SLAVE**, or **MS_UNBINDABLE** (all available since Linux 2.6.15), then the propagation type of an existing mount is changed. If more than one of these flags is specified, an error results.

The only other flags that can be specified while changing the propagation type are **MS_REC** (described below) and **MS_SILENT** (which is ignored).

The *source*, *filesystemtype*, and *data* arguments are ignored.

The meanings of the propagation type flags are as follows:

MS_SHARED

Make this mount shared. Mount and unmount events immediately under this mount will propagate to the other mounts that are members of this mount's peer group. Propagation here means that the same mount or unmount will automatically occur under all of the other mounts in the peer group. Conversely, mount and unmount events that take place under peer mounts will propagate to this mount.

MS_PRIVATE

Make this mount private. Mount and unmount events do not propagate into or out of this mount.

MS_SLAVE

If this is a shared mount that is a member of a peer group that contains other members, convert it to a slave mount. If this is a shared mount that is a member of a peer group that contains no other members, convert it to a private mount. Otherwise, the propagation type of the mount is left unchanged.

When a mount is a slave, mount and unmount events propagate into this mount from the (master) shared peer group of which it was formerly a member. Mount and unmount events under this mount do not propagate to any peer.

A mount can be the slave of another peer group while at the same time sharing mount and unmount events with a peer group of which it is a member.

MS_UNBINDABLE

Make this mount unbindable. This is like a private mount, and in addition this mount can't be bind mounted. When a recursive bind mount (**mount()** with the **MS_BIND** and **MS_REC** flags) is performed on a directory subtree, any unbindable mounts within the subtree are automatically pruned (i.e., not replicated) when replicating that subtree to produce the target subtree.

By default, changing the propagation type affects only the *target* mount. If the **MS_REC** flag is also specified in *mountflags*, then the propagation type of all mounts under *target* is also changed.

For further details regarding mount propagation types (including the default propagation type assigned to new mounts), see **mount_namespaces(7)**.

Moving a mount

If *mountflags* contains the flag **MS_MOVE** (available since Linux 2.4.18), then move a subtree: *source* specifies an existing mount and *target* specifies the new location to which that mount is to be relocated. The move is atomic: at no point is the subtree unmounted.

The remaining bits in the *mountflags* argument are ignored, as are the *filesystemtype* and *data* arguments.

Creating a new mount

If none of **MS_REMOUNT**, **MS_BIND**, **MS_MOVE**, **MS_SHARED**, **MS_PRIVATE**, **MS_SLAVE**, or **MS_UNBINDABLE** is specified in *mountflags*, then **mount()** performs its default action: creating a new mount. *source* specifies the source for the new mount, and *target* specifies the directory at which to create the mount point.

The *filesystemtype* and *data* arguments are employed, and further bits may be specified in *mountflags* to modify the behavior of the call.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

The error values given below result from filesystem type independent errors. Each filesystem type may have its own special errors and its own special behavior. See the Linux kernel source code for details.

EACCES

A component of a path was not searchable. (See also **path_resolution(7)**.)

EACCES

Mounting a read-only filesystem was attempted without giving the **MS_RDONLY** flag.

The filesystem may be read-only for various reasons, including: it resides on a read-only optical disk; it is resides on a device with a physical switch that has been set to mark the device read-only; the filesystem implementation was compiled with read-only support; or errors were detected when initially mounting the filesystem, so that it was marked read-only and can't be remounted as read-write (until the errors are fixed).

Some filesystems instead return the error **EROFS** on an attempt to mount a read-only filesystem.

EACCES

The block device *source* is located on a filesystem mounted with the **MS_NODEV** option.

EBUSY

An attempt was made to stack a new mount directly on top of an existing mount point that was created in this mount namespace with the same *source* and *target*.

EBUSY

source cannot be remounted read-only, because it still holds files open for writing.

EFAULT

One of the pointer arguments points outside the user address space.

EINVAL

source had an invalid superblock.

EINVAL

A remount operation (**MS_REMOUNT**) was attempted, but *source* was not already mounted on *target*.

EINVAL

A move operation (**MS_MOVE**) was attempted, but the mount tree under *source* includes unbindable mounts and *target* is a mount that has propagation type **MS_SHARED**.

EINVAL

A move operation (**MS_MOVE**) was attempted, but the parent mount of *source* mount has propagation type **MS_SHARED**.

EINVAL

A move operation (**MS_MOVE**) was attempted, but *source* was not a mount, or was '/'.

EINVAL

A bind operation (**MS_BIND**) was requested where *source* referred a mount namespace magic link (i.e., a */proc/[pid]/ns/mnt* magic link or a bind mount to such a link) and the propagation type of the parent mount of *target* was **MS_SHARED**, but propagation of the requested bind mount

could lead to a circular dependency that might prevent the mount namespace from ever being freed.

EINVAL

mountflags includes more than one of **MS_SHARED**, **MS_PRIVATE**, **MS_SLAVE**, or **MS_UNBINDABLE**.

EINVAL

mountflags includes **MS_SHARED**, **MS_PRIVATE**, **MS_SLAVE**, or **MS_UNBINDABLE** and also includes a flag other than **MS_REC** or **MS_SILENT**.

EINVAL

An attempt was made to bind mount an unbindable mount.

EINVAL

In an unprivileged mount namespace (i.e., a mount namespace owned by a user namespace that was created by an unprivileged user), a bind mount operation (**MS_BIND**) was attempted without specifying (**MS_REC**), which would have revealed the filesystem tree underneath one of the sub-mounts of the directory being bound.

ELOOP

Too many links encountered during pathname resolution.

ELOOP

A move operation was attempted, and *target* is a descendant of *source*.

EMFILE

(In case no block device is required:) Table of dummy devices is full.

ENAMETOOLONG

A pathname was longer than **MAXPATHLEN**.

ENODEV

filesystemtype not configured in the kernel.

ENOENT

A pathname was empty or had a nonexistent component.

ENOMEM

The kernel could not allocate a free page to copy filenames or data into.

ENOTBLK

source is not a block device (and a device was required).

ENOTDIR

target, or a prefix of *source*, is not a directory.

ENXIO

The major number of the block device *source* is out of range.

EPERM

The caller does not have the required privileges.

EPERM

An attempt was made to modify (**MS_REMOUNT**) the **MS_RDONLY**, **MS_NOSUID**, or **MS_NOEXEC** flag, or one of the "atime" flags (**MS_NOATIME**, **MS_NODIRATIME**, **MS_RELATIME**) of an existing mount, but the mount is locked; see **mount_namespaces(7)**.

EROFS

Mounting a read-only filesystem was attempted without giving the **MS_RDONLY** flag. See **EACCES**, above.

VERSIONS

The definitions of **MS_DIRSYNC**, **MS_MOVE**, **MS_PRIVATE**, **MS_REC**, **MS_RELATIME**, **MS_SHARED**, **MS_SLAVE**, **MS_STRICTATIME**, and **MS_UNBINDABLE** were added to glibc

headers in glibc 2.12.

STANDARDS

This function is Linux-specific and should not be used in programs intended to be portable.

NOTES

Since Linux 2.4 a single filesystem can be mounted at multiple mount points, and multiple mounts can be stacked on the same mount point.

The *mountflags* argument may have the magic number 0xC0ED (**MS_MGC_VAL**) in the top 16 bits. (All of the other flags discussed in DESCRIPTION occupy the low order 16 bits of *mountflags*.) Specifying **MS_MGC_VAL** was required before Linux 2.4, but since Linux 2.4 is no longer required and is ignored if specified.

The original **MS_SYNC** flag was renamed **MS_SYNCHRONOUS** in 1.1.69 when a different **MS_SYNC** was added to <*mman.h*>.

Before Linux 2.4 an attempt to execute a set-user-ID or set-group-ID program on a filesystem mounted with **MS_NOSUID** would fail with **EPERM**. Since Linux 2.4 the set-user-ID and set-group-ID bits are just silently ignored in this case.

Mount namespaces

Starting with Linux 2.4.19, Linux provides mount namespaces. A mount namespace is the set of filesystem mounts that are visible to a process. Mount namespaces can be (and usually are) shared between multiple processes, and changes to the namespace (i.e., mounts and unmounts) by one process are visible to all other processes sharing the same namespace. (The pre-2.4.19 Linux situation can be considered as one in which a single namespace was shared by every process on the system.)

A child process created by **fork(2)** shares its parent's mount namespace; the mount namespace is preserved across an **execve(2)**.

A process can obtain a private mount namespace if: it was created using the **clone(2)** **CLONE_NEWNS** flag, in which case its new namespace is initialized to be a *copy* of the namespace of the process that called **clone(2)**; or it calls **unshare(2)** with the **CLONE_NEWNS** flag, which causes the caller's mount namespace to obtain a private copy of the namespace that it was previously sharing with other processes, so that future mounts and unmounts by the caller are invisible to other processes (except child processes that the caller subsequently creates) and vice versa.

For further details on mount namespaces, see **mount_namespaces(7)**.

Parental relationship between mounts

Each mount has a parent mount. The overall parental relationship of all mounts defines the single directory hierarchy seen by the processes within a mount namespace.

The parent of a new mount is defined when the mount is created. In the usual case, the parent of a new mount is the mount of the filesystem containing the directory or file at which the new mount is attached. In the case where a new mount is stacked on top of an existing mount, the parent of the new mount is the previous mount that was stacked at that location.

The parental relationship between mounts can be discovered via the */proc/[pid]/mountinfo* file (see below).

/proc/[pid]/mounts and /proc/[pid]/mountinfo

The Linux-specific */proc/[pid]/mounts* file exposes the list of mounts in the mount namespace of the process with the specified ID. The */proc/[pid]/mountinfo* file exposes even more information about mounts, including the propagation type and mount ID information that makes it possible to discover the parental relationship between mounts. See **proc(5)** and **mount_namespaces(7)** for details of this file.

SEE ALSO

mountpoint(1), **chroot(2)**, **ioctl_iflags(2)**, **mount_setattr(2)**, **pivot_root(2)**, **umount(2)**, **mount_namespaces(7)**, **path_resolution(7)**, **findmnt(8)**, **lsblk(8)**, **mount(8)**, **umount(8)**

NAME

mount – mount a filesystem

SYNOPSIS

mount [**-h**|**-V**]

mount [**-l**] [**-t** *fstype*]

mount **-a** [**-fFnrvw**] [**-t** *fstype*] [**-O** *optlist*]

mount [**-fnrvw**] [**-o** *options*] *device|mountpoint*

mount [**-fnrvw**] [**-t** *fstype*] [**-o** *options*] *device mountpoint*

mount **--bind|--rbind|--move** *olddir newdir*

mount **--make-[shared|slave|private|unbindable|rshared|rslave|rprivate|runbindable]** *mountpoint*

DESCRIPTION

All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The **mount** command serves to attach the filesystem found on some device to the big file tree. Conversely, the **umount(8)** command will detach it again. The filesystem is used to control how data is stored on the device or provided in a virtual way by network or other services.

The standard form of the **mount** command is:

mount **-t** *type* *device dir*

This tells the kernel to attach the filesystem found on *device* (which is of type *type*) at the directory *dir*. The option **-t** *type* is optional. The **mount** command is usually able to detect a filesystem. The root permissions are necessary to mount a filesystem by default. See section "Non-superuser mounts" below for more details. The previous contents (if any) and owner and mode of *dir* become invisible, and as long as this filesystem remains mounted, the pathname *dir* refers to the root of the filesystem on *device*.

If only the directory or the device is given, for example:

mount */dir*

then **mount** looks for a mountpoint (and if not found then for a device) in the */etc/fstab* file. It's possible to use the **--target** or **--source** options to avoid ambiguous interpretation of the given argument. For example:

mount **--target** */mountpoint*

The same filesystem may be mounted more than once, and in some cases (e.g., network filesystems) the same filesystem may be mounted on the same mountpoint multiple times. The **mount** command does not implement any policy to control this behavior. All behavior is controlled by the kernel and it is usually specific to the filesystem driver. The exception is **--all**, in this case already mounted filesystems are ignored (see **--all** below for more details).

Listing the mounts

The listing mode is maintained for backward compatibility only.

For more robust and customizable output use **findmnt(8)**, especially in your scripts. Note that control characters in the mountpoint name are replaced with '?'.

The following command lists all mounted filesystems (of type *type*):

```
mount [-l] [-t type]
```

The option **-l** adds labels to this listing. See below.

Indicating the device and filesystem

Most devices are indicated by a filename (of a block special device), like */dev/sda1*, but there are other possibilities. For example, in the case of an NFS mount, *device* may look like *knuth.cwi.nl:/dir*.

The device names of disk partitions are unstable; hardware reconfiguration, and adding or removing a device can cause changes in names. This is the reason why it's strongly recommended to use filesystem or partition identifiers like UUID or LABEL. Currently supported identifiers (tags):

LABEL=*label*

Human readable filesystem identifier. See also **-L**.

UUID=*uuid*

Filesystem universally unique identifier. The format of the UUID is usually a series of hex digits separated by hyphens. See also **-U**.

Note that **mount** uses UUIDs as strings. The UUIDs from the command line or from **fstab(5)** are not converted to internal binary representation. The string representation of the UUID should be based on lower case characters.

PARTLABEL=*label*

Human readable partition identifier. This identifier is independent on filesystem and does not change by **mkfs** or **mkswap** operations. It's supported for example for GUID Partition Tables (GPT).

PARTUUID=*uuid*

Partition universally unique identifier. This identifier is independent on filesystem and does not change by **mkfs** or **mkswap** operations. It's supported for example for GUID Partition Tables (GPT).

ID=*id*

Hardware block device ID as generated by udevd. This identifier is usually based on WWN (unique storage identifier) and assigned by the hardware manufacturer. See **ls /dev/disk/by-id** for more details, this directory and running udevd is required. This identifier is not recommended for generic use as the identifier is not strictly defined and it depends on udev, udev rules and hardware.

The command **lsblk --fs** provides an overview of filesystems, LABELs and UUIDs on available block devices. The command **blkid -p <device>** provides details about a filesystem on the specified device.

Don't forget that there is no guarantee that UUIDs and labels are really unique, especially if you move, share or copy the device. Use **lsblk -o +UUID,PARTUUID** to verify that the UUIDs are really unique in your system.

The recommended setup is to use tags (e.g. **UUID=***uuid*) rather than */dev/disk/by-{label,uuid,id,partuuid,partlabel}* udev symlinks in the */etc/fstab* file. Tags are more readable, robust and portable. The **mount(8)** command internally uses udev symlinks, so the use of symlinks in */etc/fstab* has no advantage over tags. For more details see **libblkid(3)**.

The *proc* filesystem is not associated with a special device, and when mounting it, an arbitrary keyword – for example, *proc* – can be used instead of a device specification. (The customary choice *none* is less fortunate: the error message 'none already mounted' from **mount** can be confusing.)

The files /etc/fstab, /etc/mtab and /proc/mounts

The file */etc/fstab* (see **fstab(5)**), may contain lines describing what devices are usually mounted where, using which options. The default location of the **fstab(5)** file can be overridden with the **--fstab path** command-line option (see below for more details).

The command

```
mount -a [-t type] [-O optlist]
```

(usually given in a bootscript) causes all filesystems mentioned in *fstab* (of the proper type and/or having or not having the proper options) to be mounted as indicated, except for those whose line contains the **noauto** keyword. Adding the **-F** option will make **mount** fork, so that the filesystems are mounted in parallel.

When mounting a filesystem mentioned in *fstab* or *mtab*, it suffices to specify on the command line only the device, or only the mount point.

The programs **mount** and **umount(8)** traditionally maintained a list of currently mounted filesystems in the file */etc/mtab*. The support for regular classic */etc/mtab* is completely disabled at compile time by default, because on current Linux systems it is better to make */etc/mtab* a symlink to */proc/mounts* instead. The regular *mtab* file maintained in userspace cannot reliably work with namespaces, containers and other advanced Linux features. If the regular *mtab* support is enabled, then it's possible to use the file as well as the symlink.

If no arguments are given to **mount**, the list of mounted filesystems is printed.

If you want to override mount options from */etc/fstab*, you have to use the **-o** option:

```
mount device|dir -o options
```

and then the mount options from the command line will be appended to the list of options from */etc/fstab*. This default behaviour can be changed using the **--options-mode** command-line option. The usual behavior is that the last option wins if there are conflicting ones.

The **mount** program does not read the */etc/fstab* file if both *device* (or LABEL, UUID, ID, PARTUUID or PARTLABEL) and *dir* are specified. For example, to mount device **foo** at **/dir**:

```
mount /dev/foo /dir
```

This default behaviour can be changed by using the **--options-source-force** command-line option to always read configuration from *fstab*. For non-root users **mount** always reads the *fstab* configuration.

Non-superuser mounts

Normally, only the superuser can mount filesystems. However, when *fstab* contains the **user** option on a line, anybody can mount the corresponding filesystem.

Thus, given a line

```
/dev/cdrom /cd iso9660 ro,user,noauto,unhide
```

any user can mount the iso9660 filesystem found on an inserted CDROM using the command:

```
mount /cd
```

Note that **mount** is very strict about non-root users and all paths specified on command line are verified before *fstab* is parsed or a helper program is executed. It's strongly recommended to use a valid mountpoint

to specify filesystem, otherwise **mount** may fail. For example it's a bad idea to use NFS or CIFS source on command line.

Since util-linux 2.35, **mount** does not exit when user permissions are inadequate according to libmount's internal security rules. Instead, it drops uid permissions and continues as regular non-root user. This behavior supports use-cases where root permissions are not necessary (e.g., fuse filesystems, user namespaces, etc).

For more details, see **fstab(5)**. Only the user that mounted a filesystem can umount it again. If any user should be able to unmount it, then use **users** instead of **user** in the *fstab* line. The **owner** option is similar to the **user** option, with the restriction that the user must be the owner of the special file. This may be useful e.g. for */dev/fd* if a login script makes the console user owner of this device. The **group** option is similar, with the restriction that the user must be a member of the group of the special file.

Bind mount operation

Remount part of the file hierarchy somewhere else. The call is:

```
mount --bind olddir newdir
```

or by using this *fstab* entry:

```
/olddir /newdir none bind
```

After this call the same contents are accessible in two places.

It is important to understand that "bind" does not create any second-class or special node in the kernel VFS. The "bind" is just another operation to attach a filesystem. There is nowhere stored information that the filesystem has been attached by a "bind" operation. The *olddir* and *newdir* are independent and the *olddir* may be unmounted.

One can also remount a single file (on a single file). It's also possible to use a bind mount to create a mountpoint from a regular directory, for example:

```
mount --bind foo foo
```

The bind mount call attaches only (part of) a single filesystem, not possible submounts. The entire file hierarchy including submounts can be attached a second place by using:

```
mount --rbind olddir newdir
```

Note that the filesystem mount options maintained by the kernel will remain the same as those on the original mount point. The userspace mount options (e.g., *_netdev*) will not be copied by **mount** and it's necessary to explicitly specify the options on the **mount** command line.

Since util-linux 2.27 **mount** permits changing the mount options by passing the relevant options along with **--bind**. For example:

```
mount -o bind,ro foo foo
```

This feature is not supported by the Linux kernel; it is implemented in userspace by an additional **mount(2)** remounting system call. This solution is not atomic.

The alternative (classic) way to create a read-only bind mount is to use the remount operation, for example:

```
mount --bind olddir newdir
```

```
mount -o remount,bind,ro olddir newdir
```

Note that a read-only bind will create a read-only mountpoint (VFS entry), but the original filesystem superblock will still be writable, meaning that the *olddir* will be writable, but the *newdir* will be read-only.

It's also possible to change nosuid, nodev, noexec, noatime, nodiratime, relatime and nosymfollow VFS entry flags via a "remount,bind" operation. The other flags (for example filesystem-specific flags) are silently ignored. It's impossible to change mount options recursively (for example with **-o rbind,ro**).

Since util-linux 2.31, **mount** ignores the **bind** flag from */etc/fstab* on a **remount** operation (if **-o remount** is specified on command line). This is necessary to fully control mount options on remount by command line. In previous versions the bind flag has been always applied and it was impossible to re-define mount options without interaction with the bind semantic. This **mount** behavior does not affect situations when "remount,bind" is specified in the */etc/fstab* file.

The move operation

Move a **mounted tree** to another place (atomically). The call is:

```
mount --move olddir newdir
```

This will cause the contents which previously appeared under *olddir* to now be accessible under *newdir*. The physical location of the files is not changed. Note that *olddir* has to be a mountpoint.

Note also that moving a mount residing under a shared mount is invalid and unsupported. Use **findmnt -o TARGET,PROPAGATION** to see the current propagation flags.

Shared subtree operations

Since Linux 2.6.15 it is possible to mark a mount and its submounts as shared, private, slave or unbindable. A shared mount provides the ability to create mirrors of that mount such that mounts and unmounts within any of the mirrors propagate to the other mirror. A slave mount receives propagation from its master, but not vice versa. A private mount carries no propagation abilities. An unbindable mount is a private mount which cannot be cloned through a bind operation. The detailed semantics are documented in *Documentation/filesystems/sharedsubtree.txt* file in the kernel source tree; see also **mount_namespaces(7)**.

Supported operations are:

```
mount --make-shared mountpoint
mount --make-slave mountpoint
mount --make-private mountpoint
mount --make-unbindable mountpoint
```

The following commands allow one to recursively change the type of all the mounts under a given mountpoint.

```
mount --make-rshared mountpoint
mount --make-rslave mountpoint
mount --make-rprivate mountpoint
mount --make-runbindable mountpoint
```

mount does not read **fstab(5)** when a **--make-*** operation is requested. All necessary information has to be specified on the command line.

Note that the Linux kernel does not allow changing multiple propagation flags with a single **mount(2)** system call, and the flags cannot be mixed with other mount options and operations.

Since util-linux 2.23 the **mount** command can be used to do more propagation (topology) changes by one

mount(8) call and do it also together with other mount operations. The propagation flags are applied by additional **mount(2)** system calls when the preceding mount operations were successful. Note that this use case is not atomic. It is possible to specify the propagation flags in **fstab(5)** as mount options (**private**, **slave**, **shared**, **unbindable**, **rprivate**, **rslave**, **rshared**, **runbindable**).

For example:

```
mount --make-private --make-unbindable /dev/sda1 /foo
```

is the same as:

```
mount /dev/sda1 /foo
mount --make-private /foo
mount --make-unbindable /foo
```

COMMAND-LINE OPTIONS

The full set of mount options used by an invocation of **mount** is determined by first extracting the mount options for the filesystem from the *fstab* table, then applying any options specified by the **-o** argument, and finally applying a **-r** or **-w** option, when present.

The **mount** command does not pass all command-line options to the */sbin/mount.suffix* mount helpers. The interface between **mount** and the mount helpers is described below in the **EXTERNAL HELPERS** section.

Command-line options available for the **mount** command are:

-a, --all

Mount all filesystems (of the given types) mentioned in *fstab* (except for those whose line contains the **noauto** keyword). The filesystems are mounted following their order in *fstab*. The **mount** command compares filesystem source, target (and fs root for bind mount or btrfs) to detect already mounted filesystems. The kernel table with already mounted filesystems is cached during **mount --all**. This means that all duplicated *fstab* entries will be mounted.

The correct functionality depends on */proc* (to detect already mounted filesystems) and on */sys* (to evaluate filesystem tags like **UUID=** or **LABEL=**). It's strongly recommended to mount */proc* and */sys* filesystems before **mount -a** is executed, or keep */proc* and */sys* at the beginning of *fstab*.

The option **--all** is possible to use for remount operation too. In this case all filters (**-t** and **-O**) are applied to the table of already mounted filesystems.

Since version 2.35 it is possible to use the command line option **-o** to alter mount options from *fstab* (see also **--options-mode**).

Note that it is a bad practice to use **mount -a** for *fstab* checking. The recommended solution is **findmnt --verify**.

-B, --bind

Remount a subtree somewhere else (so that its contents are available in both places). See above, under **Bind mounts**.

-c, --no-canonicalize

Don't canonicalize paths. The **mount** command canonicalizes all paths (from the command line or *fstab*) by default. This option can be used together with the **-f** flag for already canonicalized absolute paths. The option is designed for mount helpers which call **mount -i**. It is strongly recommended to not use this command-line option for normal mount operations.

Note that **mount** does not pass this option to the */sbin/mount.type* helpers.

-F, --fork

(Used in conjunction with **-a**.) Fork off a new incarnation of **mount** for each device. This will do the mounts on different devices or different NFS servers in parallel. This has the advantage that it is faster; also NFS timeouts proceed in parallel. A disadvantage is that the order of the mount operations is undefined. Thus, you cannot use this option if you want to mount both */usr* and */usr/spool*.

-f, --fake

Causes everything to be done except for the actual system call; if it's not obvious, this "fakes" mounting the filesystem. This option is useful in conjunction with the **-v** flag to determine what the **mount** command is trying to do. It can also be used to add entries for devices that were mounted earlier with the **-n** option. The **-f** option checks for an existing record in */etc/mtab* and fails when the record already exists (with a regular non-fake mount, this check is done by the kernel).

-i, --internal-only

Don't call the */sbin/mount.filesystem* helper even if it exists.

-L, --label *label*

Mount the partition that has the specified *label*.

-l, --show-labels

Add the labels in the mount output. **mount** must have permission to read the disk device (e.g. be set-user-ID root) for this to work. One can set such a label for ext2, ext3 or ext4 using the **e2label(8)** utility, or for XFS using **xfs_admin(8)**, or for reiserfs using **reiserfstune(8)**.

-M, --move

Move a subtree to some other place. See above, the subsection **The move operation**.

-m, --mkdir[=*mode*]

Allow to make a target directory (mountpoint) if it does not exist yet. Alias to "**-o X-mount.mkdir[=*mode*]**", the default mode is 0755. For more details see **X-mount.mkdir** below.

-n, --no-*mtab*

Mount without writing in */etc/mtab*. This is necessary for example when */etc* is on a read-only filesystem.

-N, --namespace *ns*

Perform the mount operation in the mount namespace specified by *ns*. *ns* is either PID of process running in that namespace or special file representing that namespace.

mount switches to the mount namespace when it reads */etc/fstab*, writes */etc/mtab*: (*or writes to /run/mount*) and calls **mount(2)**, otherwise it runs in the original mount namespace. This means that the target namespace does not have to contain any libraries or other requirements necessary to execute the **mount(2)** call.

See **mount_namespaces(7)** for more information.

-O, --test-opts *opts*

Limit the set of filesystems to which the **-a** option applies. In this regard it is like the **-t** option except that **-O** is useless without **-a**. For example, the command

mount -a -O no_netdev

mounts all filesystems except those which have the option *netdev* specified in the options field in the */etc/fstab* file.

It is different from **-t** in that each option is matched exactly; a leading **no** at the beginning of one option does not negate the rest.

The **-t** and **-O** options are cumulative in effect; that is, the command

```
mount -a -t ext2 -O _netdev
```

mounts all ext2 filesystems with the *_netdev* option, not all filesystems that are either ext2 or have the *_netdev* option specified.

-o, --options *opts*

Use the specified mount options. The *opts* argument is a comma-separated list. For example:

```
mount LABEL=mydisk -o noatime,nodev,nosuid
```

For more details, see the **FILESYSTEM-INDEPENDENT MOUNT OPTIONS** and **FILESYSTEM-SPECIFIC MOUNT OPTIONS** sections.

--options-mode *mode*

Controls how to combine options from *fstab/mtab* with options from the command line. *mode* can be one of **ignore**, **append**, **prepend** or **replace**. For example, **append** means that options from *fstab* are appended to options from the command line. The default value is **prepend** — it means command line options are evaluated after *fstab* options. Note that the last option wins if there are conflicting ones.

--options-source *source*

Source of default options. *source* is a comma-separated list of **fstab**, **mtab** and **disable**. **disable** disables **fstab** and **mtab** and enables **--options-source-force**. The default value is **fstab,mtab**.

--options-source-force

Use options from *fstab/mtab* even if both *device* and *dir* are specified.

-R, --rbind

Remount a subtree and all possible submounts somewhere else (so that its contents are available in both places). See above, the subsection **Bind mounts**.

-r, --read-only

Mount the filesystem read-only. A synonym is **-o ro**.

Note that, depending on the filesystem type, state and kernel behavior, the system may still write to the device. For example, ext3 and ext4 will replay the journal if the filesystem is dirty. To prevent this kind of write access, you may want to mount an ext3 or ext4 filesystem with the **ro,noload** mount options or set the block device itself to read-only mode, see the **blockdev(8)** command.

-s

Tolerate sloppy mount options rather than failing. This will ignore mount options not supported by a filesystem type. Not all filesystems support this option. Currently it's supported by the **mount.nfs** mount helper only.

--source *device*

If only one argument for the **mount** command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is

the mount source.

--target *directory*

If only one argument for the mount command is given, then the argument might be interpreted as the target (mountpoint) or source (device). This option allows you to explicitly define that the argument is the mount target.

--target-prefix *directory*

Prepend the specified directory to all mount targets. This option can be used to follow *fstab*, but mount operations are done in another place, for example:

```
mount --all --target-prefix /chroot -o X-mount.mkdir
```

mounts all from system *fstab* to */chroot*, all missing mountpoint are created (due to *X-mount.mkdir*). See also **--fstab** to use an alternative *fstab*.

-T, --fstab *path*

Specifies an alternative *fstab* file. If *path* is a directory, then the files in the directory are sorted by **strverscmp(3)**; files that start with "." or without an *.fstab* extension are ignored. The option can be specified more than once. This option is mostly designed for initramfs or chroot scripts where additional configuration is specified beyond standard system configuration.

Note that **mount** does not pass the option **--fstab** to the **/sbin/mount.type** helpers, meaning that the alternative *fstab* files will be invisible for the helpers. This is no problem for normal mounts, but user (non-root) mounts always require *fstab* to verify the user's rights.

-t, --types *fstype*

The argument following the **-t** is used to indicate the filesystem type. The filesystem types which are currently supported depend on the running kernel. See */proc/filesystems* and */lib/modules/\$(uname -r)/kernel/fs* for a complete list of the filesystems. The most common are ext2, ext3, ext4, xfs, btrfs, vfat, sysfs, proc, nfs and cifs.

The programs **mount** and **umount(8)** support filesystem subtypes. The subtype is defined by a '.subtype' suffix. For example 'fuse.sshfs'. It's recommended to use subtype notation rather than add any prefix to the mount source (for example 'sshfs#example.com' is deprecated).

If no **-t** option is given, or if the **auto** type is specified, **mount** will try to guess the desired type. **mount** uses the **libblkid(3)** library for guessing the filesystem type; if that does not turn up anything that looks familiar, **mount** will try to read the file */etc/filesystems*, or, if that does not exist, */proc/filesystems*. All of the filesystem types listed there will be tried, except for those that are labeled "nodev" (e.g. *devpts*, *proc* and *nfs*). If */etc/filesystems* ends in a line with a single *, mount will read */proc/filesystems* afterwards. While trying, all filesystem types will be mounted with the mount option **silent**.

The **auto** type may be useful for user-mounted floppies. Creating a file */etc/filesystems* can be useful to change the probe order (e.g., to try vfat before msdos or ext3 before ext2) or if you use a kernel module autoloader.

More than one type may be specified in a comma-separated list, for the **-t** option as well as in an */etc/fstab* entry. The list of filesystem types for the **-t** option can be prefixed with **no** to specify the filesystem types on which no action should be taken. The prefix **no** has no effect when specified in an */etc/fstab* entry.

The prefix **no** can be meaningful with the **-a** option. For example, the command

mount -a -t nomsdos,smbfs

mounts all filesystems except those of type *msdos* and *smbfs*.

For most types all the **mount** program has to do is issue a simple **mount(2)** system call, and no detailed knowledge of the filesystem type is required. For a few types however (like nfs, nfs4, cifs, smbfs, ncpfs) an ad hoc code is necessary. The nfs, nfs4, cifs, smbfs, and ncpfs filesystems have a separate mount program. In order to make it possible to treat all types in a uniform way, **mount** will execute the program */sbin/mount.type* (if that exists) when called with type *type*. Since different versions of the **smbmount** program have different calling conventions, */sbin/mount.smbfs* may have to be a shell script that sets up the desired call.

-U, --uuid *uuid*

Mount the partition that has the specified *uuid*.

-v, --verbose

Verbose mode.

-w, --rw, --read-write

Mount the filesystem read/write. Read–write is the kernel default and the **mount** default is to try read–only if the previous **mount(2)** syscall with read–write flags on write–protected devices failed.

A synonym is **-o rw**.

Note that specifying **-w** on the command line forces **mount** to never try read–only mount on write–protected devices or already mounted read–only filesystems.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

FILESYSTEM-INDEPENDENT MOUNT OPTIONS

Some of these options are only useful when they appear in the */etc/fstab* file.

Some of these options could be enabled or disabled by default in the system kernel. To check the current setting see the options in */proc/mounts*. Note that filesystems also have per–filesystem specific default mount options (see for example **tune2fs -l** output for extN filesystems).

The following options apply to any filesystem that is being mounted (but not every filesystem actually honors them – e.g., the **sync** option today has an effect only for ext2, ext3, ext4, fat, vfat, ufs and xfs):

async

All I/O to the filesystem should be done asynchronously. (See also the **sync** option.)

atime

Do not use the **noatime** feature, so the inode access time is controlled by kernel defaults. See also the descriptions of the **relatime** and **strictatime** mount options.

noatime

Do not update inode access times on this filesystem (e.g. for faster access on the news spool to speed up news servers). This works for all inode types (directories too), so it implies **nodiratime**.

auto

Can be mounted with the **-a** option.

noauto

Can only be mounted explicitly (i.e., the **-a** option will not cause the filesystem to be mounted).

context=*context*, **fscontext=***context*, **defcontext=***context*, and **rootcontext=***context*

The **context=** option is useful when mounting filesystems that do not support extended attributes, such as a floppy or hard disk formatted with VFAT, or systems that are not normally running under SELinux, such as an ext3 or ext4 formatted disk from a non-SELinux workstation. You can also use **context=** on filesystems you do not trust, such as a floppy. It also helps in compatibility with xattr-supporting filesystems on earlier 2.4.<x> kernel versions. Even where xattrs are supported, you can save time not having to label every file by assigning the entire disk one security context.

A commonly used option for removable media is **context="system_u:object_r:removable_t"**.

The **fscontext=** option works for all filesystems, regardless of their xattr support. The fscontext option sets the overarching filesystem label to a specific security context. This filesystem label is separate from the individual labels on the files. It represents the entire filesystem for certain kinds of permission checks, such as during mount or file creation. Individual file labels are still obtained from the xattrs on the files themselves. The context option actually sets the aggregate context that fscontext provides, in addition to supplying the same label for individual files.

You can set the default security context for unlabeled files using **defcontext=** option. This overrides the value set for unlabeled files in the policy and requires a filesystem that supports xattr labeling.

The **rootcontext=** option allows you to explicitly label the root inode of a FS being mounted before that FS or inode becomes visible to userspace. This was found to be useful for things like stateless Linux.

Note that the kernel rejects any remount request that includes the context option, **even** when unchanged from the current context.

Warning: the *context* value might contain commas, in which case the value has to be properly quoted, otherwise **mount** will interpret the comma as a separator between mount options. Don't forget that the shell strips off quotes and thus **double quoting is required**. For example:

```
mount -t tmpfs none /mnt -o \'context="system_u:object_r:tmp_t:s0:c127,c456",noexec'
```

For more details, see **selinux(8)**.

defaults

Use the default options: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, and **async**.

Note that the real set of all default mount options depends on the kernel and filesystem type. See the beginning of this section for more details.

dev

Interpret character or block special devices on the filesystem.

nodev

Do not interpret character or block special devices on the filesystem.

diratime

Update directory inode access times on this filesystem. This is the default. (This option is ignored

when **noatime** is set.)

nodiratime

Do not update directory inode access times on this filesystem. (This option is implied when **noatime** is set.)

dirsync

All directory updates within the filesystem should be done synchronously. This affects the following system calls: **creat(2)**, **link(2)**, **unlink(2)**, **symlink(2)**, **mkdir(2)**, **rmdir(2)**, **mknod(2)** and **rename(2)**.

exec

Permit execution of binaries and other executable files.

noexec

Do not permit direct execution of any binaries on the mounted filesystem.

group

Allow an ordinary user to mount the filesystem if one of that user's groups matches the group of the device. This option implies the options **nosuid** and **nodev** (unless overridden by subsequent options, as in the option line **group,dev,suid**).

iversion

Every time the inode is modified, the *i_version* field will be incremented.

noiversion

Do not increment the *i_version* inode field.

mand

Allow mandatory locks on this filesystem. See **fctl(2)**. This option was deprecated in Linux 5.15.

nomand

Do not allow mandatory locks on this filesystem.

_netdev

The filesystem resides on a device that requires network access (used to prevent the system from attempting to mount these filesystems until the network has been enabled on the system).

nofail

Do not report errors for this device if it does not exist.

relatime

Update inode access times relative to modify or change time. Access time is only updated if the previous access time was earlier than the current modify or change time. (Similar to **noatime**, but it doesn't break **mutt(1)** or other applications that need to know if a file has been read since the last time it was modified.)

Since Linux 2.6.30, the kernel defaults to the behavior provided by this option (unless **noatime** was specified), and the **strictatime** option is required to obtain traditional semantics. In addition, since Linux 2.6.30, the file's last access time is always updated if it is more than 1 day old.

norelatime

Do not use the **relatime** feature. See also the **strictatime** mount option.

strictatime

Allows to explicitly request full atime updates. This makes it possible for the kernel to default to **relatime** or **noatime** but still allow userspace to override it. For more details about the default system mount options see */proc/mounts*.

nostrictatime

Use the kernel's default behavior for inode access time updates.

lazytime

Only update times (atime, mtime, ctime) on the in-memory version of the file inode.

This mount option significantly reduces writes to the inode table for workloads that perform frequent random writes to preallocated files.

The on-disk timestamps are updated only when:

- the inode needs to be updated for some change unrelated to file timestamps
- the application employs **fsync(2)**, **syncfs(2)**, or **sync(2)**
- an undeleted inode is evicted from memory
- more than 24 hours have passed since the inode was written to disk.

nolazytime

Do not use the lazytime feature.

suid

Honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem.

nosuid

Do not honor set-user-ID and set-group-ID bits or file capabilities when executing programs from this filesystem. In addition, SELinux domain transitions require permission *nosuid_transition*, which in turn needs also policy capability *nnp_nosuid_transition*.

silent

Turn on the silent flag.

loud

Turn off the silent flag.

owner

Allow an ordinary user to mount the filesystem if that user is the owner of the device. This option implies the options **nosuid** and **nodev** (unless overridden by subsequent options, as in the option line **owner,dev,suid**).

remount

Attempt to remount an already-mounted filesystem. This is commonly used to change the mount flags for a filesystem, especially to make a readonly filesystem writable. It does not change device or mount point.

The remount operation together with the **bind** flag has special semantics. See above, the subsection **Bind mounts**.

The remount functionality follows the standard way the **mount** command works with options from *fstab*. This means that **mount** does not read *fstab* (or *mtab*) only when both *device* and *dir* are

specified.

mount -o remount,rw /dev/foo /dir

After this call all old mount options are replaced and arbitrary stuff from *fstab* (or *mtab*) is ignored, except the **loop=** option which is internally generated and maintained by the **mount** command.

mount -o remount,rw /dir

After this call, **mount** reads *fstab* and merges these options with the options from the command line (**-o**). If no mountpoint is found in *fstab*, then a remount with unspecified source is allowed.

mount allows the use of **--all** to remount all already mounted filesystems which match a specified filter (**-O** and **-t**). For example:

mount --all -o remount,ro -t vfat

remounts all already mounted vfat filesystems in read-only mode. Each of the filesystems is remounted by **mount -o remount,ro /dir** semantic. This means the **mount** command reads *fstab* or *mtab* and merges these options with the options from the command line.

ro

Mount the filesystem read-only.

rw

Mount the filesystem read-write.

sync

All I/O to the filesystem should be done synchronously. In the case of media with a limited number of write cycles (e.g. some flash drives), **sync** may cause life-cycle shortening.

user

Allow an ordinary user to mount the filesystem. The name of the mounting user is written to the *mtab* file (or to the private libmount file in */run/mount* on systems without a regular *mtab*) so that this same user can umount the filesystem again. This option implies the options **noexec**, **nosuid**, and **nodev** (unless overridden by subsequent options, as in the option line **user,exec,dev,suid**).

nouser

Forbid an ordinary user to mount the filesystem. This is the default; it does not imply any other options.

users

Allow any user to mount and to umount the filesystem, even when some other ordinary user mounted it. This option implies the options **noexec**, **nosuid**, and **nodev** (unless overridden by subsequent options, as in the option line **users,exec,dev,suid**).

X-*

All options prefixed with "X-" are interpreted as comments or as userspace application-specific options. These options are not stored in user space (e.g., *mtab* file), nor sent to the *mount.type* helpers nor to the **mount(2)** system call. The suggested format is **X-appname.option**.

x-*

The same as **X-*** options, but stored permanently in user space. This means the options are also available for **umount(8)** or other operations. Note that maintaining mount options in user space is

tricky, because it's necessary use libmount-based tools and there is no guarantee that the options will be always available (for example after a move mount operation or in unshared namespace).

Note that before util-linux v2.30 the x-* options have not been maintained by libmount and stored in user space (functionality was the same as for X-* now), but due to the growing number of use-cases (in initrd, systemd etc.) the functionality has been extended to keep existing *fstab* configurations usable without a change.

X-mount.mkdir[=mode]

Allow to make a target directory (mountpoint) if it does not exist yet. The optional argument *mode* specifies the filesystem access mode used for **mkdir(2)** in octal notation. The default mode is 0755. This functionality is supported only for root users or when **mount** is executed without *suid* permissions. The option is also supported as **x-mount.mkdir**, but this notation is deprecated since v2.30. See also **--mkdir** command line option.

X-mount.subdir=directory

Allow mounting sub-directory from a filesystem instead of the root directory. For now, this feature is implemented by temporary filesystem root directory mount in unshared namespace and then bind the sub-directory to the final mount point and umount the root of the filesystem. The sub-directory mount shows up atomically for the rest of the system although it is implemented by multiple **mount(2)** syscalls. This feature is EXPERIMENTAL.

nosymfollow

Do not follow symlinks when resolving paths. Symlinks can still be created, and **readlink(1)**, **readlink(2)**, **realpath(1)**, and **realpath(3)** all still work properly.

FILESYSTEM-SPECIFIC MOUNT OPTIONS

This section lists options that are specific to particular filesystems. Where possible, you should first consult filesystem-specific manual pages for details. Some of those pages are listed in the following table.

Filesystem(s)	Manual page
btrfs	btrfs(5)
cifs	mount.cifs(8)
ext2, ext3, ext4	ext4(5)
fuse	fuse(8)
nfs	nfs(5)
tmpfs	tmpfs(5)
xfs	xfs(5)

Note that some of the pages listed above might be available only after you install the respective userland tools.

The following options apply only to certain filesystems. We sort them by filesystem. All options follow the **-o** flag.

What options are supported depends a bit on the running kernel. Further information may be available in filesystem-specific files in the kernel source subdirectory *Documentation/filesystems*.

Mount options for adfs

uid=value and **gid=value**

Set the owner and group of the files in the filesystem (default: uid=gid=0).

ownmask=value and **othmask=value**

Set the permission mask for ADFS 'owner' permissions and 'other' permissions, respectively (default: 0700 and 0077, respectively). See also */usr/src/linux/Documentation/filesystems/adfs.rst*.

Mount options for affs

uid=value and **gid=value**

Set the owner and group of the root of the filesystem (default: uid=gid=0, but with option **uid** or **gid** without specified value, the UID and GID of the current process are taken).

setuid=value and **setgid=value**

Set the owner and group of all files.

mode=value

Set the mode of all files to *value* & 0777 disregarding the original permissions. Add search permission to directories that have read permission. The value is given in octal.

protect

Do not allow any changes to the protection bits on the filesystem.

usemp

Set UID and GID of the root of the filesystem to the UID and GID of the mount point upon the first sync or umount, and then clear this option. Strange...

verbose

Print an informational message for each successful mount.

prefix=string

Prefix used before volume name, when following a link.

volume=string

Prefix (of length at most 30) used before '/' when following a symbolic link.

reserved=value

(Default: 2.) Number of unused blocks at the start of the device.

root=value

Give explicitly the location of the root block.

bs=value

Give blocksize. Allowed values are 512, 1024, 2048, 4096.

grpquota|noquota|quota|usrquota

These options are accepted but ignored. (However, quota utilities may react to such strings in */etc/fstab*.)

Mount options for debugfs

The debugfs filesystem is a pseudo filesystem, traditionally mounted on */sys/kernel/debug*. As of kernel version 3.4, debugfs has the following options:

uid=n, gid=n

Set the owner and group of the mountpoint.

mode=value

Sets the mode of the mountpoint.

Mount options for devpts

The devpts filesystem is a pseudo filesystem, traditionally mounted on */dev/pts*. In order to acquire a pseudo terminal, a process opens */dev/ptmx*; the number of the pseudo terminal is then made available to the process and the pseudo terminal slave can be accessed as */dev/pts/<number>*.

uid=value and gid=value

This sets the owner or the group of newly created pseudo terminals to the specified values. When nothing is specified, they will be set to the UID and GID of the creating process. For example, if there is a tty group with GID 5, then **gid=5** will cause newly created pseudo terminals to belong to the tty group.

mode=value

Set the mode of newly created pseudo terminals to the specified value. The default is 0600. A value of **mode=620** and **gid=5** makes "mesg y" the default on newly created pseudo terminals.

newinstance

Create a private instance of the devpts filesystem, such that indices of pseudo terminals allocated in this new instance are independent of indices created in other instances of devpts.

All mounts of devpts without this **newinstance** option share the same set of pseudo terminal indices (i.e., legacy mode). Each mount of devpts with the **newinstance** option has a private set of pseudo terminal indices.

This option is mainly used to support containers in the Linux kernel. It is implemented in Linux kernel versions starting with 2.6.29. Further, this mount option is valid only if **CONFIG_DEVPTS_MULTIPLE_INSTANCES** is enabled in the kernel configuration.

To use this option effectively, */dev/ptmx* must be a symbolic link to *pts/ptmx*. See *Documentation/filesystems/devpts.txt* in the Linux kernel source tree for details.

ptmxmode=value

Set the mode for the new *ptmx* device node in the devpts filesystem.

With the support for multiple instances of devpts (see **newinstance** option above), each instance has a private *ptmx* node in the root of the devpts filesystem (typically */dev/pts/ptmx*).

For compatibility with older versions of the kernel, the default mode of the new *ptmx* node is 0000. **ptmxmode=value** specifies a more useful mode for the *ptmx* node and is highly recommended when the **newinstance** option is specified.

This option is only implemented in Linux kernel versions starting with 2.6.29. Further, this option is valid only if **CONFIG_DEVPTS_MULTIPLE_INSTANCES** is enabled in the kernel configuration.

Mount options for fat

(Note: *fat* is not a separate filesystem, but a common part of the *msdos*, *umsdos* and *vfat* filesystems.)

blocksize={512|1024|2048}

Set blocksize (default 512). This option is obsolete.

uid=value and gid=value

Set the owner and group of all files. (Default: the UID and GID of the current process.)

umask=value

Set the umask (the bitmask of the permissions that are **not** present). The default is the umask of the current process. The value is given in octal.

dmask=value

Set the umask applied to directories only. The default is the umask of the current process. The value is given in octal.

fmask=value

Set the umask applied to regular files only. The default is the umask of the current process. The value is given in octal.

allow_utime=value

This option controls the permission check of mtime/atime.

20

If current process is in group of file's group ID, you can change timestamp.

2

Other users can change timestamp.

The default is set from 'dmask' option. (If the directory is writable, **utime**(2) is also allowed. I.e. ~dmask & 022)

Normally **utime**(2) checks that the current process is owner of the file, or that it has the **CAP_FOWNER** capability. But FAT filesystems don't have UID/GID on disk, so the normal check is too inflexible. With this option you can relax it.

check=value

Three different levels of pickiness can be chosen:

r[elaxed]

Upper and lower case are accepted and equivalent, long name parts are truncated (e.g. *verylongname.foobar* becomes *verylong.foo*), leading and embedded spaces are accepted in each name part (name and extension).

n[ormal]

Like "relaxed", but many special characters (*, ?, <, spaces, etc.) are rejected. This is the default.

s[trict]

Like "normal", but names that contain long parts or special characters that are sometimes used on Linux but are not accepted by MS-DOS (+, =, etc.) are rejected.

codepage=value

Sets the codepage for converting to shortname characters on FAT and VFAT filesystems. By default, codepage 437 is used.

conv=mode

This option is obsolete and may fail or be ignored.

cvf_format=module

Forces the driver to use the CVF (Compressed Volume File) module *cvf_module* instead of

auto-detection. If the kernel supports **kmod**, the **cvf_format=xxx** option also controls on-demand CVF module loading. This option is obsolete.

cvf_option=option

Option passed to the CVF module. This option is obsolete.

debug

Turn on the *debug* flag. A version string and a list of filesystem parameters will be printed (these data are also printed if the parameters appear to be inconsistent).

discard

If set, causes discard/TRIM commands to be issued to the block device when blocks are freed. This is useful for SSD devices and sparse/thinly-provisioned LUNs.

dos1xfloppy

If set, use a fallback default BIOS Parameter Block configuration, determined by backing device size. These static parameters match defaults assumed by DOS 1.x for 160 kiB, 180 kiB, 320 kiB, and 360 kiB floppies and floppy images.

errors={panic|continue|remount-ro}

Specify FAT behavior on critical errors: panic, continue without doing anything, or remount the partition in read-only mode (default behavior).

fat={12|16|32}

Specify a 12, 16 or 32 bit fat. This overrides the automatic FAT type detection routine. Use with caution!

iocharset=value

Character set to use for converting between 8 bit characters and 16 bit Unicode characters. The default is iso8859-1. Long filenames are stored on disk in Unicode format.

nfs={stale_rw|nostale_ro}

Enable this only if you want to export the FAT filesystem over NFS.

stale_rw: This option maintains an index (cache) of directory inodes which is used by the nfs-related code to improve look-ups. Full file operations (read/write) over NFS are supported but with cache eviction at NFS server, this could result in spurious **ESTALE** errors.

nostale_ro: This option bases the inode number and file handle on the on-disk location of a file in the FAT directory entry. This ensures that **ESTALE** will not be returned after a file is evicted from the inode cache. However, it means that operations such as rename, create and unlink could cause file handles that previously pointed at one file to point at a different file, potentially causing data corruption. For this reason, this option also mounts the filesystem readonly.

To maintain backward compatibility, **-o nfs** is also accepted, defaulting to **stale_rw**.

tz=UTC

This option disables the conversion of timestamps between local time (as used by Windows on FAT) and UTC (which Linux uses internally). This is particularly useful when mounting devices (like digital cameras) that are set to UTC in order to avoid the pitfalls of local time.

time_offset=minutes

Set offset for conversion of timestamps from local time used by FAT to UTC. I.e., *minutes* will be subtracted from each timestamp to convert it to UTC used internally by Linux. This is useful when the

time zone set in the kernel via **settimeofday**(2) is not the time zone used by the filesystem. Note that this option still does not provide correct time stamps in all cases in presence of DST – time stamps in a different DST setting will be off by one hour.

quiet

Turn on the *quiet* flag. Attempts to chown or chmod files do not return errors, although they fail. Use with caution!

rodir

FAT has the **ATTR_RO** (read-only) attribute. On Windows, the **ATTR_RO** of the directory will just be ignored, and is used only by applications as a flag (e.g. it's set for the customized folder).

If you want to use **ATTR_RO** as read-only flag even for the directory, set this option.

showexec

If set, the execute permission bits of the file will be allowed only if the extension part of the name is .EXE, .COM, or .BAT. Not set by default.

sys_immutable

If set, **ATTR_SYS** attribute on FAT is handled as **IMMUTABLE** flag on Linux. Not set by default.

flush

If set, the filesystem will try to flush to disk more early than normal. Not set by default.

usefree

Use the "free clusters" value stored on **FSINFO**. It'll be used to determine number of free clusters without scanning disk. But it's not used by default, because recent Windows don't update it correctly in some case. If you are sure the "free clusters" on **FSINFO** is correct, by this option you can avoid scanning disk.

dots, nodots, dotsOK=[yes|no]

Various misguided attempts to force Unix or DOS conventions onto a FAT filesystem.

Mount options for hfs**creator=cccc, type=cccc**

Set the creator/type values as shown by the MacOS finder used for creating new files. Default values: '????'.

uid=n, gid=n

Set the owner and group of all files. (Default: the UID and GID of the current process.)

dir_umask=n, file_umask=n, umask=n

Set the umask used for all directories, all regular files, or all files and directories. Defaults to the umask of the current process.

session=n

Select the CDROM session to mount. Defaults to leaving that decision to the CDROM driver. This option will fail with anything but a CDROM as underlying device.

part=n

Select partition number n from the device. Only makes sense for CDROMs. Defaults to not parsing the partition table at all.

quiet

Don't complain about invalid mount options.

Mount options for hpfs**uid=value** and **gid=value**

Set the owner and group of all files. (Default: the UID and GID of the current process.)

umask=value

Set the umask (the bitmask of the permissions that are **not** present). The default is the umask of the current process. The value is given in octal.

case={lower|asis}

Convert all files names to lower case, or leave them. (Default: **case=lower**.)

conv=mode

This option is obsolete and may fail or being ignored.

nocheck

Do not abort mounting when certain consistency checks fail.

Mount options for iso9660

ISO 9660 is a standard describing a filesystem structure to be used on CD-ROMs. (This filesystem type is also seen on some DVDs. See also the *udf* filesystem.)

Normal *iso9660* filenames appear in an 8.3 format (i.e., DOS-like restrictions on filename length), and in addition all characters are in upper case. Also there is no field for file ownership, protection, number of links, provision for block/character devices, etc.

Rock Ridge is an extension to iso9660 that provides all of these UNIX-like features. Basically there are extensions to each directory record that supply all of the additional information, and when Rock Ridge is in use, the filesystem is indistinguishable from a normal UNIX filesystem (except that it is read-only, of course).

norock

Disable the use of Rock Ridge extensions, even if available. Cf. **map**.

nojoliet

Disable the use of Microsoft Joliet extensions, even if available. Cf. **map**.

check={r[elaxed]|s[trict]}

With **check=relaxed**, a filename is first converted to lower case before doing the lookup. This is probably only meaningful together with **norock** and **map=normal**. (Default: **check=strict**.)

uid=value and **gid=value**

Give all files in the filesystem the indicated user or group id, possibly overriding the information found in the Rock Ridge extensions. (Default: **uid=0,gid=0**.)

map={n[ormal]|o[ff]|a[corn]}

For non-Rock Ridge volumes, normal name translation maps upper to lower case ASCII, drops a trailing ';', and converts ';' to '!'. With **map=off** no name translation is done. See **norock**. (Default: **map=normal**.) **map=acorn** is like **map=normal** but also apply Acorn extensions if present.

mode=value

For non-Rock Ridge volumes, give all files the indicated mode. (Default: read and execute permission for everybody.) Octal mode values require a leading 0.

unhide

Also show hidden and associated files. (If the ordinary files and the associated or hidden files have the

same filenames, this may make the ordinary files inaccessible.)

block={512|1024|2048}

Set the block size to the indicated value. (Default: **block=1024**.)

conv=mode

This option is obsolete and may fail or being ignored.

cruft

If the high byte of the file length contains other garbage, set this mount option to ignore the high order bits of the file length. This implies that a file cannot be larger than 16 MB.

session=x

Select number of session on a multisession CD.

sbsector=xxx

Session begins from sector xxx.

The following options are the same as for vfat and specifying them only makes sense when using discs encoded using Microsoft's Joliet extensions.

iocharset=value

Character set to use for converting 16 bit Unicode characters on CD to 8 bit characters. The default is iso8859-1.

utf8

Convert 16 bit Unicode characters on CD to UTF-8.

Mount options for jfs
iocharset=name

Character set to use for converting from Unicode to ASCII. The default is to do no conversion. Use **iocharset=utf8** for UTF8 translations. This requires **CONFIG_NLS_UTF8** to be set in the kernel .config file.

resize=value

Resize the volume to *value* blocks. JFS only supports growing a volume, not shrinking it. This option is only valid during a remount, when the volume is mounted read-write. The **resize** keyword with no value will grow the volume to the full size of the partition.

nointegrity

Do not write to the journal. The primary use of this option is to allow for higher performance when restoring a volume from backup media. The integrity of the volume is not guaranteed if the system abnormally ends.

integrity

Default. Commit metadata changes to the journal. Use this option to remount a volume where the **nointegrity** option was previously specified in order to restore normal behavior.

errors={continue|remount-ro|panic}

Define the behavior when an error is encountered. (Either ignore errors and just mark the filesystem erroneous and continue, or remount the filesystem read-only, or panic and halt the system.)

noquota|quota|usrquota|grpquota

These options are accepted but ignored.

Mount options for msdos

See mount options for fat. If the *msdos* filesystem detects an inconsistency, it reports an error and sets the file system read-only. The filesystem can be made writable again by remounting it.

Mount options for ncpfs

Just like *nfs*, the *ncpfs* implementation expects a binary argument (a *struct ncp_mount_data*) to the **mount**(2) system call. This argument is constructed by **ncpmount**(8) and the current version of **mount** (2.12) does not know anything about *ncpfs*.

Mount options for ntfs**iocharset=name**

Character set to use when returning file names. Unlike VFAT, NTFS suppresses names that contain nonconvertible characters. Deprecated.

nls=name

New name for the option earlier called *iocharset*.

utf8

Use UTF-8 for converting file names.

uni_xlate={0|1|2}

For 0 (or 'no' or 'false'), do not use escape sequences for unknown Unicode characters. For 1 (or 'yes' or 'true') or 2, use vfat-style 4-byte escape sequences starting with ":". Here 2 gives a little-endian encoding and 1 a byteswapped bigendian encoding.

posix=[0|1]

If enabled (posix=1), the filesystem distinguishes between upper and lower case. The 8.3 alias names are presented as hard links instead of being suppressed. This option is obsolete.

uid=value, gid=value and umask=value

Set the file permission on the filesystem. The umask value is given in octal. By default, the files are owned by root and not readable by somebody else.

Mount options for overlay

Since Linux 3.18 the overlay pseudo filesystem implements a union mount for other filesystems.

An overlay filesystem combines two filesystems – an **upper** filesystem and a **lower** filesystem. When a name exists in both filesystems, the object in the upper filesystem is visible while the object in the lower filesystem is either hidden or, in the case of directories, merged with the upper object.

The lower filesystem can be any filesystem supported by Linux and does not need to be writable. The lower filesystem can even be another overlayfs. The upper filesystem will normally be writable and if it is it must support the creation of trusted.* extended attributes, and must provide a valid d_type in readdir responses, so NFS is not suitable.

A read-only overlay of two read-only filesystems may use any filesystem type. The options **lowerdir** and **upperdir** are combined into a merged directory by using:

```
mount -t overlay overlay \
    -olowerdir=/lower,upperdir=/upper,workdir=/work /merged
```

lowerdir=directory

Any filesystem, does not need to be on a writable filesystem.

upperdir=directory

The upperdir is normally on a writable filesystem.

workdir=directory

The workdir needs to be an empty directory on the same filesystem as upperdir.

userxattr

Use the "**user.overlay.**" xattr namespace instead of "**trusted.overlay.**". This is useful for unprivileged mounting of overlays.

redirect_dir={on|off|follow|nofollow}

If the *redirect_dir* feature is enabled, then the directory will be copied up (but not the contents). Then the "**{trusted|user}.overlay.redirect**" extended attribute is set to the path of the original location from the root of the overlay. Finally the directory is moved to the new location.

on

Redirects are enabled.

off

Redirects are not created and only followed if "redirect_always_follow" feature is enabled in the kernel/module config.

follow

Redirects are not created, but followed.

nofollow

Redirects are not created and not followed (equivalent to "redirect_dir=off" if "redirect_always_follow" feature is not enabled).

index={on|off}

Inode index. If this feature is disabled and a file with multiple hard links is copied up, then this will "break" the link. Changes will not be propagated to other names referring to the same inode.

uuid={on|off}

Can be used to replace UUID of the underlying filesystem in file handles with null, and effectively disable UUID checks. This can be useful in case the underlying disk is copied and the UUID of this copy is changed. This is only applicable if all lower/upper/work directories are on the same filesystem, otherwise it will fallback to normal behaviour.

nfs_export={on|off}

When the underlying filesystem supports NFS export and the "nfs_export" feature is enabled, an overlay filesystem may be exported to NFS.

With the "nfs_export" feature, on copy_up of any lower object, an index entry is created under the index directory. The index entry name is the hexadecimal representation of the copy up origin file handle. For a non-directory object, the index entry is a hard link to the upper inode. For a directory object, the index entry has an extended attribute "**{trusted|user}.overlay.upper**" with an encoded file handle of the upper directory inode.

When encoding a file handle from an overlay filesystem object, the following rules apply

- For a non-upper object, encode a lower file handle from lower inode
- For an indexed object, encode a lower file handle from copy_up origin
- For a pure-upper object and for an existing non-indexed upper object, encode an upper file handle from upper inode

The encoded overlay file handle includes

- Header including path type information (e.g. lower/upper)
- UUID of the underlying filesystem
- Underlying filesystem encoding of underlying inode

This encoding format is identical to the encoding format of file handles that are stored in extended attribute "`{trusted|user}.overlay.origin`". When decoding an overlay file handle, the following steps are followed

- Find underlying layer by UUID and path type information.
- Decode the underlying filesystem file handle to underlying dentry.
- For a lower file handle, lookup the handle in index directory by name.
- If a whiteout is found in index, return **ESTALE**. This represents an overlay object that was deleted after its file handle was encoded.
- For a non-directory, instantiate a disconnected overlay dentry from the decoded underlying dentry, the path type and index inode, if found.
- For a directory, use the connected underlying decoded dentry, path type and index, to lookup a connected overlay dentry.

Decoding a non-directory file handle may return a disconnected dentry. `copy_up` of that disconnected dentry will create an upper index entry with no upper alias.

When overlay filesystem has multiple lower layers, a middle layer directory may have a "redirect" to lower directory. Because middle layer "redirects" are not indexed, a lower file handle that was encoded from the "redirect" origin directory, cannot be used to find the middle or upper layer directory. Similarly, a lower file handle that was encoded from a descendant of the "redirect" origin directory, cannot be used to reconstruct a connected overlay path. To mitigate the cases of directories that cannot be decoded from a lower file handle, these directories are copied up on encode and encoded as an upper file handle. On an overlay filesystem with no upper layer this mitigation cannot be used NFS export in this setup requires turning off redirect follow (e.g. `"redirect_dir=nofollow"`).

The overlay filesystem does not support non-directory connectable file handles, so exporting with the `subtree_check` exportfs configuration will cause failures to lookup files over NFS.

When the NFS export feature is enabled, all directory index entries are verified on mount time to check that upper file handles are not stale. This verification may cause significant overhead in some cases.

Note: the mount options `index=off,nfs_export=on` are conflicting for a read-write mount and will result in an error.

xino={on|off|auto}

The "xino" feature composes a unique object identifier from the real object `st_ino` and an underlying `fsid` index. The "xino" feature uses the high inode number bits for `fsid`, because the underlying filesystems rarely use the high inode number bits. In case the underlying inode number does overflow into the high xino bits, overlay filesystem will fall back to the non xino behavior for that inode.

For a detailed description of the effect of this option please refer to
<https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html?highlight=overlayfs>

metacopy={on|off}

When metadata only copy up feature is enabled, `overlayfs` will only copy up metadata (as opposed to whole file), when a metadata specific operation like `chown/chmod` is performed. Full file will be

copied up later when file is opened for WRITE operation.

In other words, this is delayed data copy up operation and data is copied up when there is a need to actually modify data.

volatile

Volatile mounts are not guaranteed to survive a crash. It is strongly recommended that volatile mounts are only used if data written to the overlay can be recreated without significant effort.

The advantage of mounting with the "volatile" option is that all forms of sync calls to the upper filesystem are omitted.

In order to avoid giving a false sense of safety, the syncfs (and fsync) semantics of volatile mounts are slightly different than that of the rest of VFS. If any writeback error occurs on the upperdir's filesystem after a volatile mount takes place, all sync functions will return an error. Once this condition is reached, the filesystem will not recover, and every subsequent sync call will return an error, even if the upperdir has not experienced a new error since the last sync call.

When overlay is mounted with "volatile" option, the directory "\$workdir/work/incompat/volatile" is created. During next mount, overlay checks for this directory and refuses to mount if present. This is a strong indicator that user should throw away upper and work directories and create fresh one. In very limited cases where the user knows that the system has not crashed and contents of upperdir are intact, the "volatile" directory can be removed.

Mount options for reiserfs

Reiserfs is a journaling filesystem.

conv

Instructs version 3.6 reiserfs software to mount a version 3.5 filesystem, using the 3.6 format for newly created objects. This filesystem will no longer be compatible with reiserfs 3.5 tools.

hash={rupasov|tea|r5|detect}

Choose which hash function reiserfs will use to find files within directories.

rupasov

A hash invented by Yury Yu. Rupasov. It is fast and preserves locality, mapping lexicographically close file names to close hash values. This option should not be used, as it causes a high probability of hash collisions.

tea

A Davis–Meyer function implemented by Jeremy Fitzhardinge. It uses hash permuting bits in the name. It gets high randomness and, therefore, low probability of hash collisions at some CPU cost. This may be used if **EHASHCOLLISION** errors are experienced with the r5 hash.

r5

A modified version of the rupasov hash. It is used by default and is the best choice unless the filesystem has huge directories and unusual file-name patterns.

detect

Instructs **mount** to detect which hash function is in use by examining the filesystem being mounted, and to write this information into the reiserfs superblock. This is only useful on the first mount of an old format filesystem.

hashed_relocation

Tunes the block allocator. This may provide performance improvements in some situations.

no_unhashed_relocation

Tunes the block allocator. This may provide performance improvements in some situations.

noborder

Disable the border allocator algorithm invented by Yury Yu. Rupasov. This may provide performance improvements in some situations.

nolog

Disable journaling. This will provide slight performance improvements in some situations at the cost of losing reiserfs's fast recovery from crashes. Even with this option turned on, reiserfs still performs all journaling operations, save for actual writes into its journaling area. Implementation of *nolog* is a work in progress.

notail

By default, reiserfs stores small files and 'file tails' directly into its tree. This confuses some utilities such as **lilo(8)**. This option is used to disable packing of files into the tree.

replayonly

Replay the transactions which are in the journal, but do not actually mount the filesystem. Mainly used by *reiserfsck*.

resize=number

A remount option which permits online expansion of reiserfs partitions. Instructs reiserfs to assume that the device has *number* blocks. This option is designed for use with devices which are under logical volume management (LVM). There is a special *resizer* utility which can be obtained from [ftp://ftp.namesys.com/pub/reiserfsprogs](http://ftp.namesys.com/pub/reiserfsprogs).

user_xattr

Enable Extended User Attributes. See the **attr(1)** manual page.

acl

Enable POSIX Access Control Lists. See the **acl(5)** manual page.

barrier=none / barrier=flush

This disables / enables the use of write barriers in the journaling code. **barrier=none** disables, **barrier=flush** enables (default). This also requires an IO stack which can support barriers, and if reiserfs gets an error on a barrier write, it will disable barriers again with a warning. Write barriers enforce proper on-disk ordering of journal commits, making volatile disk write caches safe to use, at some performance penalty. If your disks are battery-backed in one way or another, disabling barriers may safely improve performance.

Mount options for ubifs

UBIFS is a flash filesystem which works on top of UBI volumes. Note that **atime** is not supported and is always turned off.

The device name may be specified as

ubiX_Y

UBI device number **X**, volume number **Y**

ubiY

UBI device number **0**, volume number **Y**

ubiX:NAME

UBI device number **X**, volume with name **NAME**

ubi:NAME

UBI device number **0**, volume with name **NAME**

Alternative ! separator may be used instead of ::

The following mount options are available:

bulk_read

Enable bulk-read. VFS read-ahead is disabled because it slows down the filesystem. Bulk-Read is an internal optimization. Some flashes may read faster if the data are read at one go, rather than at several read requests. For example, OneNAND can do "read-while-load" if it reads more than one NAND page.

no_bulk_read

Do not bulk-read. This is the default.

chk_data_crc

Check data CRC-32 checksums. This is the default.

no_chk_data_crc

Do not check data CRC-32 checksums. With this option, the filesystem does not check CRC-32 checksum for data, but it does check it for the internal indexing information. This option only affects reading, not writing. CRC-32 is always calculated when writing the data.

compr={none|lzo|zlib}

Select the default compressor which is used when new files are written. It is still possible to read compressed files if mounted with the **none** option.

Mount options for udf

UDF is the "Universal Disk Format" filesystem defined by OSTA, the Optical Storage Technology Association, and is often used for DVD-ROM, frequently in the form of a hybrid UDF/ISO-9660 filesystem. It is, however, perfectly usable by itself on disk drives, flash drives and other block devices. See also *iso9660*.

uid=

Make all files in the filesystem belong to the given user. **uid=forget** can be specified independently of (or usually in addition to) **uid=<user>** and results in UDF not storing uids to the media. In fact the recorded uid is the 32-bit overflow uid -1 as defined by the UDF standard. The value is given as either **<user>** which is a valid user name or the corresponding decimal user id, or the special string "forget".

gid=

Make all files in the filesystem belong to the given group. **gid=forget** can be specified independently of (or usually in addition to) **gid=<group>** and results in UDF not storing gids to the media. In fact the recorded gid is the 32-bit overflow gid -1 as defined by the UDF standard. The value is given as either **<group>** which is a valid group name or the corresponding decimal group id, or the special string "forget".

umask=

Mask out the given permissions from all inodes read from the filesystem. The value is given in octal.

mode=

If **mode=** is set the permissions of all non-directory inodes read from the filesystem will be set to the given mode. The value is given in octal.

dmode=

If **dmode**= is set the permissions of all directory inodes read from the filesystem will be set to the given dmode. The value is given in octal.

bs=

Set the block size. Default value prior to kernel version 2.6.30 was 2048. Since 2.6.30 and prior to 4.11 it was logical device block size with fallback to 2048. Since 4.11 it is logical block size with fallback to any valid block size between logical device block size and 4096.

For other details see the **mkudffs(8)** 2.0+ manpage, see the **COMPATIBILITY** and **BLOCK SIZE** sections.

unhide

Show otherwise hidden files.

undelete

Show deleted files in lists.

adinicb

Embed data in the inode. (default)

noadinicb

Don't embed data in the inode.

shortad

Use short UDF address descriptors.

longad

Use long UDF address descriptors. (default)

nostrict

Unset strict conformance.

ioccharset=

Set the NLS character set. This requires kernel compiled with **CONFIG_UDF_NLS** option.

utf8

Set the UTF-8 character set.

Mount options for debugging and disaster recovery**novrs**

Ignore the Volume Recognition Sequence and attempt to mount anyway.

session=

Select the session number for multi-session recorded optical media. (default= last session)

anchor=

Override standard anchor location. (default= 256)

lastblock=

Set the last block of the filesystem.

Unused historical mount options that may be encountered and should be removed**uid=ignore**

Ignored, use uid=<user> instead.

gid=ignore

Ignored, use `gid=<group>` instead.

volume=

Unimplemented and ignored.

partition=

Unimplemented and ignored.

fileset=

Unimplemented and ignored.

rootdir=

Unimplemented and ignored.

Mount options for ufs**ufs_{type}=*value***

UFS is a filesystem widely used in different operating systems. The problem are differences among implementations. Features of some implementations are undocumented, so its hard to recognize the type of ufs automatically. That's why the user must specify the type of ufs by mount option. Possible values are:

old

Old format of ufs, this is the default, read only. (Don't forget to give the `-r` option.)

44bsd

For filesystems created by a BSD-like system (NetBSD, FreeBSD, OpenBSD).

ufs2

Used in FreeBSD 5.x supported as read-write.

5xbsd

Synonym for ufs2.

sun

For filesystems created by SunOS or Solaris on Sparc.

sunx86

For filesystems created by Solaris on x86.

hp

For filesystems created by HP-UX, read-only.

nextstep

For filesystems created by NeXTStep (on NeXT station) (currently read only).

nextstep-cd

For NextStep CDROMs (`block_size == 2048`), read-only.

openstep

For filesystems created by OpenStep (currently read only). The same filesystem type is also used by macOS.

onerror=*value*

Set behavior on error:

panic

If an error is encountered, cause a kernel panic.

[lock|umount|repair]

These mount options don't do anything at present; when an error is encountered only a console message is printed.

Mount options for umsdos

See mount options for msdos. The **dotsOK** option is explicitly killed by *umsdos*.

Mount options for vfat

First of all, the mount options for *fat* are recognized. The **dotsOK** option is explicitly killed by *vfat*. Furthermore, there are

uni_xlate

Translate unhandled Unicode characters to special escaped sequences. This lets you backup and restore filenames that are created with any Unicode characters. Without this option, a '?' is used when no translation is possible. The escape character is ':' because it is otherwise invalid on the vfat filesystem. The escape sequence that gets used, where u is the Unicode character, is: ':', (u & 0x3f), ((u>>6) & 0x3f), (u>>12).

posix

Allow two files with names that only differ in case. This option is obsolete.

nonumtail

First try to make a short name without sequence number, before trying *name~num.ext*.

utf8

UTF8 is the filesystem safe 8-bit encoding of Unicode that is used by the console. It can be enabled for the filesystem with this option or disabled with utf8=0, utf8=no or utf8=false. If *uni_xlate* gets set, UTF8 gets disabled.

shortname=*mode*

Defines the behavior for creation and display of filenames which fit into 8.3 characters. If a long name for a file exists, it will always be the preferred one for display. There are four *modes*:

lower

Force the short name to lower case upon display; store a long name when the short name is not all upper case.

win95

Force the short name to upper case upon display; store a long name when the short name is not all upper case.

winnt

Display the short name as is; store a long name when the short name is not all lower case or all upper case.

mixed

Display the short name as is; store a long name when the short name is not all upper case. This mode is the default since Linux 2.6.32.

Mount options for usbfs**devuid=*uid* and devgid=*gid* and devmode=*mode***

Set the owner and group and mode of the device files in the usbfs filesystem (default: uid=gid=0, mode=0644). The mode is given in octal.

busuid=uid and **busgid=gid** and **busmode=mode**

Set the owner and group and mode of the bus directories in the usbfss filesystem (default: uid=gid=0, mode=0555). The mode is given in octal.

listuid=uid and **listgid=gid** and **listmode=mode**

Set the owner and group and mode of the file *devices* (default: uid=gid=0, mode=0444). The mode is given in octal.

DM-VERITY SUPPORT

The device-mapper verity target provides read-only transparent integrity checking of block devices using kernel crypto API. The **mount** command can open the dm-verity device and do the integrity verification before the device filesystem is mounted. Requires libcryptsetup with in libmount (optionally via **dlopen(3)**). If libcryptsetup supports extracting the root hash of an already mounted device, existing devices will be automatically reused in case of a match. Mount options for dm-verity:

verity.hashdevice=path

Path to the hash tree device associated with the source volume to pass to dm-verity.

verity.roothash=hex

Hex-encoded hash of the root of *verity.hashdevice*. Mutually exclusive with *verity.roothashfile*.

verity.roothashfile=path

Path to file containing the hex-encoded hash of the root of *verity.hashdevice*. Mutually exclusive with *verity.roothash*.

verity.hashoffset=offset

If the hash tree device is embedded in the source volume, *offset* (default: 0) is used by dm-verity to get to the tree.

verity.fecdevice=path

Path to the Forward Error Correction (FEC) device associated with the source volume to pass to dm-verity. Optional. Requires kernel built with **CONFIG_DM_VERITY_FEC**.

verity.fecoffset=offset

If the FEC device is embedded in the source volume, *offset* (default: 0) is used by dm-verity to get to the FEC area. Optional.

verity.fecroots=value

Parity bytes for FEC (default: 2). Optional.

verity.roothashsig=path

Path to **pkes7(1ssl)** signature of root hash hex string. Requires **crypt_activate_by_signed_key()** from cryptsetup and kernel built with **CONFIG_DM_VERITY_VERIFY_ROOTHASH_SIG**. For device reuse, signatures have to be either used by all mounts of a device or by none. Optional.

verity.oncorruption=ignore|restart|panic

Instruct the kernel to ignore, reboot or panic when corruption is detected. By default the I/O operation simply fails. Requires Linux 4.1 or newer, and libcryptsetup 2.3.4 or newer. Optional.

Supported since util-linux v2.35.

For example commands:

```
mksquashfs /etc /tmp/etc.squashfs
dd if=/dev/zero of=/tmp/etc.hash bs=1M count=10
```

```
veritysetup format /tmp/etc.squashfs /tmp/etc.hash
openssl smime -sign -in <hash> -nocerts -inkey private.key \
-signer private.crt -noattr -binary -outform der -out /tmp/etc.roothash.p7s
mount -o verity.hashdevice=/tmp/etc.hash,verity.roothash=<hash>, \
verity.roothashsig=/tmp/etc.roothash.p7s /tmp/etc.squashfs /mnt
```

create squashfs image from */etc* directory, verity hash device and mount verified filesystem image to */mnt*. The kernel will verify that the root hash is signed by a key from the kernel keyring if roothashsig is used.

LOOP-DEVICE SUPPORT

One further possible type is a mount via the loop device. For example, the command

```
mount /tmp/disk.img /mnt -t vfat -o loop=dev/loop3
```

will set up the loop device */dev/loop3* to correspond to the file */tmp/disk.img*, and then mount this device on */mnt*.

If no explicit loop device is mentioned (but just an option '**-o loop**' is given), then **mount** will try to find some unused loop device and use that, for example

```
mount /tmp/disk.img /mnt -o loop
```

The **mount** command **automatically** creates a loop device from a regular file if a filesystem type is not specified or the filesystem is known for libblkid, for example:

```
mount /tmp/disk.img /mnt
```

```
mount -t ext4 /tmp/disk.img /mnt
```

This type of mount knows about three options, namely **loop**, **offset** and **sizelimit**, that are really options to **losetup(8)**. (These options can be used in addition to those specific to the filesystem type.)

Since Linux 2.6.25 auto-destruction of loop devices is supported, meaning that any loop device allocated by **mount** will be freed by **umount** independently of */etc/mtab*.

You can also free a loop device by hand, using **losetup -d** or **umount -d**.

Since util-linux v2.29, **mount** re-uses the loop device rather than initializing a new device if the same backing file is already used for some loop device with the same offset and sizelimit. This is necessary to avoid a filesystem corruption.

EXIT STATUS

mount has the following exit status values (the bits can be ORed):

0
success

1
incorrect invocation or permissions

2
system error (out of memory, cannot fork, no more loop devices)

4
internal **mount** bug

8

user interrupt

16problems writing or locking */etc/mtab***32**

mount failure

64

some mount succeeded

The command **mount -a** returns 0 (all succeeded), 32 (all failed), or 64 (some failed, some succeeded).

EXTERNAL HELPERS

The syntax of external mount helpers is:

/sbin/mount.*suffix* *spec dir* [−sfnv**] [−**N namespace**] [−**o options**] [−**t type.subtype**]**

where the *suffix* is the filesystem type and the **−sfnvN** options have the same meaning as the normal mount options. The **−t** option is used for filesystems with subtypes support (for example **/sbin/mount.fuse −t fuse.sshfs**).

The command **mount** does not pass the mount options **unbindable**, **runbindable**, **private**, **rprivate**, **slave**, **rslave**, **shared**, **rshared**, **auto**, **noauto**, **comment**, **x−***, **loop**, **offset** and **sizelimit** to the mount.<*suffix*> helpers. All other options are used in a comma-separated list as an argument to the **−o** option.

ENVIRONMENT

LIBMOUNT_FSTAB=<path>

overrides the default location of the *fstab* file (ignored for suid)

LIBMOUNT_MTAB=<path>

overrides the default location of the *mtab* file (ignored for suid)

LIBMOUNT_DEBUG=all

enables libmount debug output

LIBBLKID_DEBUG=all

enables libblkid debug output

LOOPDEV_DEBUG=all

enables loop device setup debug output

FILES

See also "The files */etc/fstab*, */etc/mtab* and */proc/mounts*" section above.

/etc/fstab

filesystem table

/run/mount

libmount private runtime directory

/etc/mtab

table of mounted filesystems or symlink to */proc/mounts*

/etc/mtab~

lock file (unused on systems with *mtab* symlink)

/etc/mtab.tmp

temporary file (unused on systems with *mtab* symlink)

/etc/filesystems

a list of filesystem types to try

HISTORY

A **mount** command existed in Version 5 AT&T UNIX.

BUGS

It is possible for a corrupted filesystem to cause a crash.

Some Linux filesystems don't support **-o sync** and **-o dirsync** (the ext2, ext3, ext4, fat and vfat filesystems do support synchronous updates (a la BSD) when mounted with the **sync** option).

The **-o remount** may not be able to change mount parameters (all *ext2fs*-specific parameters, except **sb**, are changeable with a remount, for example, but you can't change **gid** or **umask** for the *fatfs*).

It is possible that the files */etc/mtab* and */proc/mounts* don't match on systems with a regular *mtab* file. The first file is based only on the **mount** command options, but the content of the second file also depends on the kernel and others settings (e.g. on a remote NFS server — in certain cases the **mount** command may report unreliable information about an NFS mount point and the */proc/mount* file usually contains more reliable information.) This is another reason to replace the *mtab* file with a symlink to the */proc/mounts* file.

Checking files on NFS filesystems referenced by file descriptors (i.e. the **fcntl** and **ioctl** families of functions) may lead to inconsistent results due to the lack of a consistency check in the kernel even if the **noac** mount option is used.

The **loop** option with the **offset** or **sizelimit** options used may fail when using older kernels if the **mount** command can't confirm that the size of the block device has been configured as requested. This situation can be worked around by using the **losetup(8)** command manually before calling **mount** with the configured loop device.

AUTHORS

Karel Zak <kzak@redhat.com>

SEE ALSO

mount(2), **umount(2)**, **filesystems(5)**, **fstab(5)**, **nfs(5)**, **xfs(5)**, **mount_namespaces(7)**, **xattr(7)**, **e2label(8)**, **findmnt(8)**, **losetup(8)**, **lsblk(8)**, **mke2fs(8)**, **mountd(8)**, **nfsd(8)**, **swapon(8)**, **tune2fs(8)**, **umount(8)**, **xfs_admin(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **mount** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

fuse – configuration and mount options for FUSE file systems

DESCRIPTION

FUSE (Filesystem in Userspace) is a simple interface for userspace programs to export a virtual filesystem to the Linux kernel. FUSE also aims to provide a secure method for non privileged users to create and mount their own filesystem implementations.

DEFINITIONS

FUSE The in-kernel filesystem that forwards requests to a user-space process.

filesystem

The user-space process that responds to requests received from the kernel.

libfuse The shared library that most (user-space) filesystems use to communicate with FUSE (the kernel filesystem). libfuse also provides the **fusermount3** (or **fusermount** if you have older version of libfuse) helper to allow non-privileged users to mount filesystems.

filesystem owner

The user that starts the filesystem and instructs the kernel to associate it with a particular mount-point. The latter is typically done by the filesystem itself on start-up. When using libfuse, this is done by calling the **fusermount3** utility.

client Any process that interacts with the mountpoint.

CONFIGURATION

Some options regarding mount policy can be set in the file */etc/fuse.conf*. Currently these options are:

mount_max = NNN

Set the maximum number of FUSE mounts allowed to non-root users. The default is 1000.

user_allow_other

Allow non-root users to specify the **allow_other** or **allow_root** mount options (see below).

These limits are enforced by the **fusermount3** helper, so they can be avoided by filesystems that run as root.

OPTIONS

Most of the generic mount options described in **mount** are supported (**ro**, **rw**, **suid**, **nosuid**, **dev**, **nodev**, **exec**, **noexec**, **atime**, **noatime**, **sync**, **async**, **dirsync**). Filesystems are mounted with **nodev,nosuid** by default, which can only be overridden by a privileged user.

General mount options:

These are FUSE specific mount options that can be specified for all filesystems:

default_permissions

This option instructs the kernel to perform its own permission check instead of deferring all permission checking to the filesystem. The check by the kernel is done in addition to any permission checks by the filesystem, and both have to succeed for an operation to be allowed. The kernel performs a standard UNIX permission check (based on mode bits and ownership of the directory entry, and uid/gid of the client).

This mount option is activated implicitly if the filesystem enables ACL support during the initial feature negotiation when opening the device fd. In this case, the kernel performs both ACL and standard unix permission checking.

Filesystems that do not implement any permission checking should generally add this option internally.

allow_other

This option overrides the security measure restricting file access to the filesystem owner, so that all users (including root) can access the files.

rootmode=M

Specifies the file mode of the filesystem's root (in octal representation).

blkdev Mount a filesystem backed by a block device. This is a privileged option. The device must be specified with the **fsname=NAME** option.

blksize=N

Set the block size for the filesystem. This option is only valid for 'fuseblk' type mounts. The default is 512.

In most cases, this option should not be specified by the filesystem owner but set internally by the filesystem.

max_read=N

With this option the maximum size of read operations can be set. The default is infinite, but typically the kernel enforces its own limit in addition to this one. A value of zero corresponds to no limit.

This option should not be specified by the filesystem owner. The correct (or optimum) value depends on the filesystem implementation and should thus be set by the filesystem internally.

This mount option is deprecated in favor of direct negotiation over the device fd (as done for e.g. the maximum size of write operations). For the time being, libfuse-using filesystems that want to limit the read size must therefore use this mount option *and* set the same value again in the init() handler.

fd=N The file descriptor to use for communication between the userspace filesystem and the kernel. The file descriptor must have been obtained by opening the FUSE device (/dev/fuse).

This option should not be specified by the filesystem owner. It is set by libfuse (or, if libfuse is not used, must be set by the filesystem itself).

user_id=N

group_id=N Specifies the numeric uid/gid of the mount owner.

This option should not be specified by the filesystem owner. It is set by libfuse (or, if libfuse is not used, must be set by the filesystem itself).

fsname=NAME

Sets the filesystem source (first field in /etc/mtab). The default is the name of the filesystem process.

subtype=TYPE

Sets the filesystem type (third field in /etc/mtab). The default is the name of the filesystem process. If the kernel supports it, /etc/mtab and /proc/mounts will show the filesystem type as **fuse.TYPE**

If the kernel doesn't support subtypes, the source field will be **TYPE#NAME**, or if **fsname** option is not specified, just **TYPE**.

libfuse-specific mount options:

These following options are not actually passed to the kernel but interpreted by libfuse. They can be specified for all filesystems that use libfuse:

allow_root

This option is similar to **allow_other** but file access is limited to the filesystem owner and root. This option and **allow_other** are mutually exclusive.

auto_unmount

This option enables automatic release of the mountpoint if filesystem terminates for any reason. Normally the filesystem is responsible for releasing the mountpoint, which means that the mountpoint becomes inaccessible if the filesystem terminates without first unmounting.

At the moment, this option implies that the filesystem will also be mounted with **nodev** and **nosuid** (even when mounted by root). This restriction may be lifted in the future.

High-level mount options:

These following options are not actually passed to the kernel but interpreted by libfuse. They can only be specified for filesystems that use the high-level libfuse API:

kernel_cache

This option disables flushing the cache of the file contents on every **open(2)**. This should only be enabled on filesystems, where the file data is never changed externally (not through the mounted FUSE filesystem). Thus it is not suitable for network filesystems and other "intermediate" filesystems.

NOTE: if this option is not specified (and neither **direct_io**) data is still cached after the **open(2)**, so a **read(2)** system call will not always initiate a read operation.

auto_cache

This option is an alternative to **kernel_cache**. Instead of unconditionally keeping cached data, the cached data is invalidated on **open(2)** if the modification time or the size of the file has changed since it was last opened.

umask=M

Override the permission bits in *st_mode* set by the filesystem. The resulting permission bits are the ones missing from the given umask value. The value is given in octal representation.

uid=N Override the *st_uid* field set by the filesystem (N is numeric).

gid=N Override the *st_gid* field set by the filesystem (N is numeric).

entry_timeout=T

The timeout in seconds for which name lookups will be cached. The default is 1.0 second. For all the timeout options, it is possible to give fractions of a second as well (e.g. **entry_timeout=2.8**)

negative_timeout=T

The timeout in seconds for which a negative lookup will be cached. This means, that if file did not exist (lookup returned **ENOENT**), the lookup will only be redone after the timeout, and the file/directory will be assumed to not exist until then. The default is 0.0 second, meaning that caching negative lookups are disabled.

attr_timeout=T

The timeout in seconds for which file/directory attributes are cached. The default is 1.0 second.

ac_attr_timeout=T

The timeout in seconds for which file attributes are cached for the purpose of checking if **auto_cache** should flush the file data on **open**. The default is the value of **attr_timeout**

noforget**remember=T**

Normally, libfuse assigns inodes to paths only for as long as the kernel is aware of them. With this option inodes are instead assigned for at least T seconds (or, in the case of **noforget**, the life-time of the filesystem). This will require more memory, but may be necessary when using applications that make use of inode numbers.

modules=M1[:M2...]

Add modules to the filesystem stack. Modules are pushed in the order they are specified, with the original filesystem being on the bottom of the stack.

mount.fuse3 options:

These options are interpreted by **mount.fuse3** and are thus only available when mounting a file system via **mount.fuse3** (such as when mounting via the generic **mount(1)** command or **/etc/fstab**). Supported options are:

setuid=USER

Switch to **USER** and its primary group before launching the FUSE file system process. **mount.fuse3** must be run as root or with **CAP_SETUID** and **CAP_SETGID** for this to work.

drop_privileges

Perform setup of the FUSE file descriptor and mounting the file system before launching the FUSE file system process. **mount.fuse3** requires privilege to do so, i.e. must be run as root or at least with **CAP_SYS_ADMIN** and **CAP_SETPCAP**. It will launch the file system process fully unprivileged, i.e. without **capabilities(7)** and **prctl(2)** flags set up such that privileges can't be reacquired (e.g. via setuid or fscaps binaries). This reduces risk in the event of the FUSE file system process getting compromised by malicious file system data.

FUSE MODULES (STACKING)

Modules are filesystem stacking support to high level API. Filesystem modules can be built into libfuse or loaded from shared object

iconv

Perform file name character set conversion. Options are:

from_code=CHARSET

Character set to convert from (see **iconv -l** for a list of possible values). Default is **UTF-8**.

to_code=CHARSET

Character set to convert to. Default is determined by the current locale.

subdir

Prepend a given directory to each path. Options are:

subdir=DIR

Directory to prepend to all paths. This option is *mandatory*.

rellinks

Transform absolute symlinks into relative

norellinks

Do not transform absolute symlinks into relative. This is the default.

SECURITY

The fusermount3 program is installed set-user-gid to fuse. This is done to allow users from fuse group to mount their own filesystem implementations. There must however be some limitations, in order to prevent Bad User from doing nasty things. Currently those limitations are:

1. The user can only mount on a mountpoint, for which it has write permission
2. The mountpoint is not a sticky directory which isn't owned by the user (like **/tmp** usually is)
3. No other user (including root) can access the contents of the mounted filesystem.

NOTE

FUSE filesystems are unmounted using the **fusermount3(1)** command (**fusermount3 -u mountpoint**).

AUTHORS

FUSE is currently maintained by Nikolaus Rath <Nikolaus@rath.org>

The original author of FUSE is Miklos Szeredi <mszteredi@inf.bme.hu>.

This man page was originally written by Bastien Roucaries <roucaries.bastien+debian@gmail.com> for the Debian GNU/Linux distribution.

SEE ALSO

fusermount3(1) fusermount(1) mount(8) fuse(4)

mount.nfs(8) - Linux man page

Name

mount.nfs, mount.nfs4 - mount a Network File System

Synopsis

mount.nfs *remotetarget dir [-rvVwfnsH] [-o options]*

Description

mount.nfs is a part of [**nfs**\(5\)](#) utilities package, which provides NFS client functionality.

mount.nfs is meant to be used by the [**mount**\(8\)](#) command for mounting NFS shares. This subcommand, however, can also be used as a standalone command with limited functionality.

mount.nfs4 is used for mounting NFSv4 file system, while **mount.nfs** is used to mount NFS file systems versions 3 or 2. *remotetarget* is a server share usually in the form of **servername:/path/to/share**. *dir* is the directory on which the file system is to be mounted.

Options

-r

Mount file system readonly.

-v

Be verbose.

-V

Print version.

-w

Mount file system read-write.

-f

Fake mount. Don't actually call the mount system call.

-n

Do not update */etc/mtab*. By default, an entry is created in */etc/mtab* for every mounted file system. Use this option to skip making an entry.

-s

Tolerate sloppy mount options rather than fail.

-h

Print help message.

nfsoptions

Refer to [nfs\(5\)](#) or [mount\(8\)](#) manual pages.

Note

For further information please refer [nfs\(5\)](#) and [mount\(8\)](#) manual pages.

Files

/etc/fstab

file system table

/etc/mtab

table of mounted file systems

See Also

[nfs\(5\)](#), [mount\(8\)](#),

Author

Amit Gud <agud@redhat.com>

Referenced By

[auto.master\(5\)](#)

NAME

ntfs-3g – Third Generation Read/Write NTFS Driver

SYNOPSIS

```
ntfs-3g [-o option[,...]] volume mount_point
mount -t ntfs-3g [-o option[,...]] volume mount_point
lowntfs-3g [-o option[,...]] volume mount_point
mount -t lowntfs-3g [-o option[,...]] volume mount_point
```

DESCRIPTION

ntfs-3g is an NTFS driver, which can create, remove, rename, move files, directories, hard links, and streams; it can read and write files, including streams, sparse files and transparently compressed files; it can handle special files like symbolic links, devices, and FIFOs; moreover it provides standard management of file ownership and permissions, including POSIX ACLs.

It comes in two variants **ntfs-3g** and **lowntfs-3g** with a few differences mentioned below in relevant options descriptions.

The *volume* to be mounted can be either a block device or an image file, either by using the *mount* command or starting the drive.

Windows hibernation and fast restarting

On computers which can be dual-booted into Windows or Linux, Windows has to be fully shut down before booting into Linux, otherwise the NTFS file systems on internal disks may be left in an inconsistent state and changes made by Linux may be ignored by Windows.

So, Windows may not be left in hibernation when starting Linux, in order to avoid inconsistencies. Moreover, the fast restart feature available on recent Windows systems has to be disabled. This can be achieved by issuing as an Administrator the Windows command which disables both hibernation and fast restarting :

```
powercfg /h off
```

If either Windows is hibernated or its fast restart is enabled, partitions on internal disks are forced to be mounted in read-only mode.

Access Handling and Security

By default, files and directories are owned by the effective user and group of the mounting process, and everybody has full read, write, execution and directory browsing permissions. You can also assign permissions to a single user by using the **uid** and/or the **gid** options together with the **umask**, or **fmask** and **dmask** options.

Doing so, all Windows users have full access to the files created by **ntfs-3g**.

But, by setting the **permissions** option, you can benefit from the full ownership and permissions features as defined by POSIX. Moreover, by defining a Windows-to-Linux user mapping, the ownerships and permissions are even applied to Windows users and conversely.

If **ntfs-3g** is set setuid-root then non-root users will be also able to mount volumes.

Windows Filename Compatibility

NTFS supports several filename namespaces: DOS, Win32 and POSIX. While the **ntfs-3g** driver handles all of them, it always creates new files in the POSIX namespace for maximum portability and interoperability reasons. This means that filenames are case sensitive and all characters are allowed except '/' and '\0'. This is perfectly legal on Windows, though some application may get confused. The option **windows_names** may be used to apply Windows restrictions to new file names.

Alternate Data Streams (ADS)

NTFS stores all data in streams. Every file has exactly one unnamed data stream and can have many named data streams. The size of a file is the size of its unnamed data stream. By default, **ntfs-3g** will only read the unnamed data stream.

By using the option **streams_interface=windows**, with the ntfs-3g driver (not possible with lowntfs-3g),

you will be able to read any named data streams, simply by specifying the stream name after a colon. For example:

```
cat some.mp3:artist
```

Named data streams act like normal files, so you can read from them, write to them and even delete them (using rm). You can list all the named data streams a file has by getting the **ntfs.streams.list** extended attribute.

OPTIONS

Below is a summary of the options that **ntfs-3g** accepts.

acl Enable setting Posix ACLs on created files and use them for access control. This option is only available on specific builds. It is set by default when a user mapping file is present and the **permissions** mount option is not set.

allow_other

This option overrides the security measure restricting file access to the user mounting the filesystem. This option is only allowed to root, but this restriction can be overridden by the **user_allow_other** option in the /etc/fuse.conf file.

atime, noatime, relatime

The **atime** option updates inode access time for each access.

The **noatime** option disables inode access time updates, which can speed up file operations and prevent sleeping (notebook) disks spinning up too often thus saving energy and disk lifetime.

The **relatime** option is very similar to **noatime**. It updates inode access times relative to modify or change time. The access time is only updated if the previous access time was earlier than the current modify or change time. Unlike **noatime** this option doesn't break applications that need to know if a file has been read since the last time it was modified. This is the default behaviour.

big_writes

This option prevents fuse from splitting write buffers into 4K chunks, enabling big write buffers to be transferred from the application in a single step (up to some system limit, generally 128K bytes).

compression

This option enables creating new transparently compressed files in directories marked for compression. A directory is marked for compression by setting the bit 11 (value 0x00000800) in its Windows attribute. In such a directory, new files are created compressed and new subdirectories are themselves marked for compression. The option and the flag have no effect on existing files. Currently this is the default option.

debug Makes ntfs-3g (or lowntfs-3g) to print a lot of debug output from libntfs-3g and FUSE.

delay_mtime[= value]

Only update the file modification time and the file change time of a file when it is closed or when the indicated delay since the previous update has elapsed. The argument is a number of seconds, with a default value of 60. This is mainly useful for big files which are kept open for a long time and written to without changing their size, such as databases or file system images mounted as loop.

dmask=value

Set the bitmask of the directory permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

efs_raw

This option should only be used in backup or restore situation. It changes the apparent size of files and the behavior of read and write operation so that encrypted files can be saved and restored

without being decrypted. The **user.ntfs.efsinfo** extended attribute has also to be saved and restored for the file to be decrypted.

fmask=value

Set the bitmask of the file permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

force This option is obsolete. It has been superseded by the **recover** and **norecover** options.

hide_dot_files

Set the hidden flag in the NTFS attribute for created files and directories whose first character of the name is a dot. Such files and directories normally do not appear in directory listings, and when the flag is set they do not appear in Windows directory displays either. When a file is renamed or linked with a new name, the hidden flag is adjusted to the latest name.

hide_hid_files

Hide the hidden files and directories in directory listings, the hidden files and directories being the ones whose NTFS attribute have the hidden flag set. The hidden files will not be selected when using wildcards in commands, but all files and directories remain accessible by full name, for example you can always display the Windows trash bin directory by : "ls -ld '\$RECYCLE.BIN'".

ignore_case (only with lowntfs-3g)

Ignore character case when accessing a file (**FOO**, **Foo**, **foo**, etc. designate the same file). All files are displayed with lower case in directory listings.

inherit When creating a new file, set its initial protections according to inheritance rules defined in parent directory. These rules deviate from Posix specifications, but yield a better Windows compatibility. The **permissions** (or ****acl****) option or a valid user mapping file is required for this option to be effective.

locale=value

This option can be useful when wanting a language specific locale environment. It is however discouraged as it leads to files with untranslatable characters to not be visible.

max_read=value

With this option the maximum size of read operations can be set. The default is infinite. Note that the size of read requests is limited anyway by the system (usually to 128kbyte).

no_def_opts

By default ntfs-3g acts as if **silent** (ignore permission errors when permissions are not enabled), **allow_other** (allow any user to access files) and **nonempty** (allow mounting on non-empty directories) were set, and **no_def_opts** cancels these default options.

no_detach

Makes ntfs-3g to not detach from terminal and print some debug output.

nocompression

This option disables creating new transparently compressed files in directories marked for compression. Existing compressed files can still be read and updated.

norecover

Do not try to mount a partition which was not unmounted properly by Windows.

permissions

Set standard permissions on created files and use standard access control. This option is set by default when a user mapping file is present.

posix_nlink

Compute the count of hard links of a file or directory according to the POSIX specifications. When this option is not set, a count of 1 is set for directories, and the short name of files is accounted for. Using the option entails some penalty as the count is not stored and has to be computed.

recover

Recover and try to mount a partition which was not unmounted properly by Windows. The Windows logfile is cleared, which may cause inconsistencies. Currently this is the default option.

remove_hiberfile

When the NTFS volume is hibernated, a read-write mount is denied and a read-only mount is forced. One needs either to resume Windows and shutdown it properly, or use this option which will remove the Windows hibernation file. Please note, this means that the saved Windows session will be completely lost. Use this option under your own responsibility.

- ro** Mount the filesystem read-only. Useful if Windows is hibernated or the NTFS journal file is unclean.

show_sys_files

Show the metafiles in directory listings. Otherwise the default behaviour is to hide the metafiles, which are special files used to store the NTFS structure. Please note that even when this option is specified, "\$MFT" may not be visible due to a glibc bug. Furthermore, irrespectively of **show_sys_files**, all files are accessible by name, for example you can always do "ls -l '\$UpCase'".

- silent** Do nothing, without returning any error, on chmod and chown operations and on permission checking errors, when the **permissions** option is not set and no user mapping file is defined. This option is on by default, and when set off (through option **no_def_opts**) ownership and permissions parameters have to be set.

special_files=mode

This option selects a mode for representing a special file to be created (symbolic link, socket, fifo, character or block device). The *mode* can be **interix** or **wsl**, and existing files in either mode are recognized irrespective of the selected mode. Interix is the traditional mode, used by default, and wsl is interoperable with Windows WSL, but it is not compatible with Windows versions earlier than Windows 10. Neither mode are interoperable with Windows.

streams_interface=mode

This option controls how the user can access Alternate Data Streams (ADS) or in other words, named data streams. The *mode* can be set to one of **none**, **windows** or **xattr**. If the option is set to **none**, the user will have no access to the named data streams. If it is set to **windows** (not possible with lowntfs-3g), then the user can access them just like in Windows (eg. cat file:stream). If it's set to **xattr**, then the named data streams are mapped to extended attributes and a user can manipulate them using {get,set}fattr utilities. The default is **xattr**.

uid=value and gid=value

Set the owner and the group of files and directories. The values are numerical. The defaults are the uid and gid of the current process.

umask=value

Set the bitmask of the file and directory permissions that are not present. The value is given in octal. The default value is 0 which means full access to everybody.

usermapping=file-name

Use file *file-name* as the user mapping file instead of the default **.NTFS-3G/UserMapping**. If *file-name* defines a full path, the file must be located on a partition previously mounted. If it defines a relative path, it is interpreted relative to the root of NTFS partition being mounted.

When a user mapping file is defined, the options **uid=**, **gid=**, **umask=**, **fmask=**, **dmask=** and **silent** are ignored.

user_xattr

Same as **streams_interface=xattr**.

windows_names

This option prevents files, directories and extended attributes to be created with a name not allowed by windows, because

- it contains some not allowed character,
- or the last character is a space or a dot,
- or the name is reserved.

The forbidden characters are the nine characters " * / : < > ? \ | and those whose code is less than 0x20, and the reserved names are CON, PRN, AUX, NUL, COM1..COM9, LPT1..LPT9, with no suffix or followed by a dot.

Existing such files can still be read (and renamed).

USER MAPPING

NTFS uses specific ids to record the ownership of files instead of the **uid** (user id) and **gid** (group id) used by Linux. As a consequence a mapping between the ids has to be defined for ownerships to be recorded into NTFS files and recognized.

By default, this mapping is fetched from the file **.NTFS-3G/UserMapping** located in the NTFS partition. The option **usermapping=** may be used to define another location. When the option **permissions** is set and no mapping file is found, a default mapping is used.

Each line in the user mapping file defines a mapping. It is organized in three fields separated by colons. The first field identifies a **uid**, the second field identifies a **gid** and the third one identifies the corresponding NTFS id, known as a **SID**. The **uid** and the **gid** are optional and defining both of them for the same **SID** is not recommended.

If no interoperation with Windows is needed, you can use the option **permissions** to define a standard mapping. Alternately, you may define your own mapping by setting a single default mapping with no uid and gid. In both cases, files created on Linux will appear to Windows as owned by a foreign user, and files created on Windows will appear to Linux as owned by root. Just copy the example below and replace the 9 and 10-digit numbers by any number not greater than 4294967295. The resulting behavior is the same as the one with the option **permission** set with no ownership option and no user mapping file available.

```
::S-1-5-21-3141592653-589793238-462643383-10000
```

If a strong interoperation with Windows is needed, the mapping has to be defined for each user and group known to both system, and the **SIDs** used by Windows has to be collected. This will lead to a user mapping file like :

```
john::S-1-5-21-3141592653-589793238-462643383-1008
mary::S-1-5-21-3141592653-589793238-462643383-1009
smith::S-1-5-21-3141592653-589793238-462643383-513
::S-1-5-21-3141592653-589793238-462643383-10000
```

The utility **ntfsusermap** may be used to create such a user mapping file.

EXAMPLES

Mount /dev/sda1 to /mnt/windows:

```
ntfs-3g /dev/sda1 /mnt/windows
```

or

```
mount -t ntfs-3g /dev/sda1 /mnt/windows
```

Mount the ntfs data partition /dev/sda3 to /mnt/data with standard Linux permissions applied :

```
ntfs-3g -o permissions /dev/sda3 /mnt/data
```

or

```
mount -t ntfs-3g -o permissions /dev/sda3 /mnt/data
```

Read-only mount /dev/sda5 to /home/user/mnt and make user with uid 1000 to be the owner of all files:

```
ntfs-3g /dev/sda5 /home/user/mnt -o ro,uid=1000
```

/etc/fstab entry for the above (the sixth and last field has to be zero to avoid a file system check at boot time) :

```
/dev/sda5 /home/user/mnt ntfs-3g ro,uid=1000 0 0
```

Unmount /mnt/windows:

```
umount /mnt/windows
```

EXIT CODES

To facilitate the use of the **ntfs-3g** driver in scripts, an exit code is returned to give an indication of the mountability status of a volume. Value 0 means success, and all other ones mean an error. The unique error codes are documented in the **ntfs-3g.probe(8)** manual page.

KNOWN ISSUES

Please see

<https://github.com/tuxera/ntfs-3g/wiki/NTFS-3G-FAQ>

for common questions and known issues. If you would find a new one in the latest release of the software then please post an ntfs-3g issue describing it in detail so that the development team can be aware of the issue and take care of it:

<https://github.com/tuxera/ntfs-3g/issues>

AUTHORS

ntfs-3g was based on and a major improvement to ntfsmount and libntfs which were written by Yura Pakhuchiy and the Linux-NTFS team. The improvements were made, the ntfs-3g project was initiated and currently led by long time Linux-NTFS team developer Szabolcs Szakacsits (szaka@tuxera.com).

THANKS

Several people made heroic efforts, often over five or more years which resulted the ntfs-3g driver. Most importantly they are Anton Altaparmakov, Jean-Pierre André, Erik Larsson, Richard Russon, Szabolcs Szakacsits, Yura Pakhuchiy, Yuval Fledel, and the author of the groundbreaking FUSE filesystem development framework, Miklos Szeredi.

SEE ALSO

ntfs-3g.probe(8), ntfsprogs(8), attr(5), getfattr(1)

mountd(8) - Linux man page

Name

rpc.mountd - NFS mount daemon

Synopsis

/usr/sbin/rpc.mountd [options]

Description

The **rpc.mountd** daemon implements the server side of the NFS MOUNT protocol, an NFS side protocol used by NFS version 2 [RFC1094] and NFS version 3 [RFC1813].

An NFS server maintains a table of local physical file systems that are accessible to NFS clients. Each file system in this table is referred to as an *exported file system*, or *export*, for short.

Each file system in the export table has an access control list. **rpc.mountd** uses these access control lists to determine whether an NFS client is permitted to access a given file system. For details on how to manage your NFS server's export table, see the **exports(5)** and **exportfs(8)** man pages.

Mounting exported NFS File Systems

The NFS MOUNT protocol has several procedures. The most important of these are MNT (mount an export) and UMNT (unmount an export).

A MNT request has two arguments: an explicit argument that contains the pathname of the root directory of the export to be mounted, and an implicit argument that is the sender's IP address.

When receiving a MNT request from an NFS client, **rpc.mountd** checks both the pathname and the sender's IP address against its export table. If the sender is permitted to access the requested export, **rpc.mountd** returns an NFS file handle for the export's root directory to the client. The client can then use the root file handle and NFS LOOKUP requests to navigate the directory structure of the export.

The rmtab File

The **rpc.mountd** daemon registers every successful MNT request by adding an entry to the `/var/lib/nfs/rmtab` file. When receiving a UMNT request from an NFS client, **rpc.mountd** simply removes the matching entry from `/var/lib/nfs/rmtab`, as long as the access control list for that export allows that sender to access the export.

Clients can discover the list of file systems an NFS server is currently exporting, or the list of other clients that have mounted its exports, by using the **showmount**(8) command. **showmount**(8) uses other procedures in the NFS MOUNT protocol to report information about the server's exported file systems.

Note, however, that there is little to guarantee that the contents of */var/lib/nfs/rmtab* are accurate. A client may continue accessing an export even after invoking UMNT. If the client reboots without sending a UMNT request, stale entries remain for that client in */var/lib/nfs/rmtab*.

Options

-d kind or --debug kind

Turn on debugging. Valid kinds are: all, auth, call, general and parse.

-F or --foreground

Run in foreground (do not daemonize)

-f or --exports-file

This option specifies the exports file, listing the clients that this server is prepared to serve and parameters to apply to each such mount (see **exports**(5)). By default, export information is read from */etc(exports*.

-h or --help

Display usage message.

-o num or --descriptors num

Set the limit of the number of open file descriptors to num. The default is to leave the limit unchanged.

-N or --no-nfs-version

This option can be used to request that **rpc.mountd** do not offer certain versions of NFS. The current version of **rpc.mountd** can support both NFS version 2, 3 and 4. If either one of these version should not be offered, **rpc.mountd** must be invoked with the option **--no-nfs-version <vers>** .

-n or --no-tcp

Don't advertise TCP for mount.

-P

Ignored (compatibility with unfsd??).

-p or --port num

Specifies the port number used for RPC listener sockets. If this option is not specified, **rpc.mountd** chooses a random ephemeral port for each listener socket.

This option can be used to fix the port value of **rpc.mountd**'s listeners when NFS MOUNT requests must traverse a firewall between clients and servers.

-H or --ha-callout prog

Specify a high availability callout program. This program receives callouts for all MOUNT and UNMOUNT requests. This allows **rpc.mountd** to be used in a High

Availability NFS (HA-NFS) environment.

The callout program is run with 4 arguments. The first is **mount** or **umount** depending on the reason for the callout. The second will be the name of the client performing the mount. The third will be the path that the client is mounting. The last is the number of concurrent mounts that we believe the client has of that path.

This callout is not needed with 2.6 and later kernels. Instead, mount the nfsd filesystem on `/proc/fs/nfsd`.

-s, --state-directory-path *directory*

Specify a directory in which to place statd state information. If this option is not specified the default of `/var/lib/nfs` is used.

-r, --reverse-lookup

rpc.mountd tracks IP addresses in the `rmtab` file. When a DUMP request is made (by someone running **showmount -a**, for instance), it returns IP addresses instead of hostnames by default. This option causes **rpc.mountd** to perform a reverse lookup on each IP address and return that hostname instead. Enabling this can have a substantial negative effect on performance in some situations.

-t N or **--num-threads=N**

This option specifies the number of worker threads that **rpc.mountd** spawns. The default is 1 thread, which is probably enough. More threads are usually only needed for NFS servers which need to handle mount storms of hundreds of NFS mounts in a few seconds, or when your DNS server is slow or unreliable.

-V or **--nfs-version**

This option can be used to request that **rpc.mountd** offer certain versions of NFS. The current version of **rpc.mountd** can support both NFS version 2 and the newer version 3.

-v or **--version**

Print the version of **rpc.mountd** and exit.

-g or **--manage-gids**

Accept requests from the kernel to map user id numbers into lists of group id numbers for use in access control. An NFS request will normally (except when using Kerberos or other cryptographic authentication) contains a user-id and a list of group-ids. Due to a limitation in the NFS protocol, at most 16 groups ids can be listed. If you use the **-g** flag, then the list of group ids received from the client will be replaced by a list of group ids determined by an appropriate lookup on the server. Note that the 'primary' group id is not affected so a **newgroup** command on the client will still be effective. This function requires a Linux Kernel with version at least 2.6.21.

TCP_WRAPPERS SUPPORT

You can protect your **rpc.mountd** listeners using the **tcp_wrapper** library or **iptables**(8).

Note that the **tcp_wrapper** library supports only IPv4 networking.

Add the hostnames of NFS peers that are allowed to access **rpc.mountd** to */etc/hosts.allow*. Use the daemon name **mountd** even if the **rpc.mountd** binary has a different name.

Hostnames used in either access file will be ignored when they can not be resolved into IP addresses. For further information see the **tcpd**(8) and **hosts_access**(5) man pages.

IPv6 and TI-RPC support

TI-RPC is a pre-requisite for supporting NFS on IPv6. If TI-RPC support is built into **rpc.mountd**, it attempts to start listeners on network transports marked 'visible' in */etc/netconfig*. As long as at least one network transport listener starts successfully, **rpc.mountd** will operate.

Files

/etc/exports

input file for **exportfs**, listing exports, export options, and access control lists

/var/lib/nfs/rmtab

table of clients accessing server's exports

See Also

[exportfs\(8\)](#), [exports\(5\)](#), [showmount\(8\)](#), [rpc.nfsd\(8\)](#), [rpc.rquotad\(8\)](#), [nfs\(5\)](#), [tcpd\(8\)](#), [hosts_access\(5\)](#), [iptables\(8\)](#), [netconfig\(5\)](#)

RFC 1094 - "NFS: Network File System Protocol Specification"

RFC 1813 - "NFS Version 3 Protocol Specification"

Author

Olaf Kirch, H. J. Lu, G. Allan Morris III, and a host of others.

Referenced By

[fsinfo\(8\)](#), [mount\(8\)](#), [nfsd\(7\)](#)

Name

`mshowfat` - shows FAT clusters allocated to file

Note of warning

This manpage has been automatically generated from mtools's texinfo documentation, and may not be entirely accurate or complete. See the end of this man page for details.

Description

The `mshowfat` command is used to display the FAT entries for a file. Syntax:

`mshowfat [-o offset] files`

If no offset is given, a list of all clusters occupied by the file is printed. If an offset is given, only the number of the cluster containing that offset is printed.

See Also

Mtools' texinfo doc

Viewing the texi doc

This manpage has been automatically generated from mtools's texinfo documentation. However, this process is only approximative, and some items, such as crossreferences, footnotes and indices are lost in this translation process. Indeed, these items have no appropriate representation in the manpage format. Moreover, not all information has been translated into the manpage version. Thus I strongly advise you to use the original texinfo doc. See the end of this manpage for instructions how to view the texinfo doc.

- * To generate a printable copy from the texinfo doc, run the following commands:

`./configure; make dvi; dvips mtools.dvi`

- * To generate a html copy, run:

`./configure; make html`

A premade html can be found at '<http://www.gnu.org/software/mtools/manual/mtools.html>'

- * To generate an info copy (browsable using emacs' info mode), run:

`./configure; make info`

The texinfo doc looks most pretty when printed or as html. Indeed, in the info version certain examples are difficult to read due to the quoting conventions used in info.

NAME

mv – move (rename) files

SYNOPSIS

```
mv [OPTION]... [-T] SOURCE DEST
mv [OPTION]... SOURCE... DIRECTORY
mv [OPTION]... -t DIRECTORY SOURCE...
```

DESCRIPTION

Rename *SOURCE* to *DEST*, or move *SOURCE*(s) to *DIRECTORY*.

Mandatory arguments to long options are mandatory for short options too.

--backup[=CONTROL]

make a backup of each existing destination file

-b like **--backup** but does not accept an argument

-f, --force

do not prompt before overwriting

-i, --interactive

prompt before overwrite

-n, --no-clobber

do not overwrite an existing file

If you specify more than one of **-i**, **-f**, **-n**, only the final one takes effect.

--strip-trailing-slashes

remove any trailing slashes from each *SOURCE* argument

-S, --suffix=SUFFIX

override the usual backup suffix

-t, --target-directory=DIRECTORY

move all *SOURCE* arguments into *DIRECTORY*

-T, --no-target-directory

treat *DEST* as a normal file

-u, --update

move only when the *SOURCE* file is newer than the destination file or when the destination file is missing

-v, --verbose

explain what is being done

-Z, --context

set SELinux security context of destination file to default type

--help display this help and exit

--version

output version information and exit

The backup suffix is ‘`~’’, unless set with **--suffix** or **SIMPLE_BACKUP_SUFFIX**. The version control method may be selected via the **--backup** option or through the **VERSION_CONTROL** environment variable. Here are the values:

none, off

never make backups (even if **--backup** is given)

numbered, t

make numbered backups

existing, nil
numbered if numbered backups exist, simple otherwise
simple, never
always make simple backups

AUTHOR

Written by Mike Parker, David MacKenzie, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent
permitted by law.

SEE ALSO

`rename(2)`

Full documentation <<https://www.gnu.org/software/coreutils/mv>>
or available locally via: `info '(coreutils) mv invocation'`

NAME

nano – Nano’s ANOther editor, inspired by Pico

SYNOPSIS

nano [*options*] [[+*line[,column]*] *file*]...

nano [*options*] [[+[**crCR**](/|?)*string*] *file*]...

DESCRIPTION

nano is a small and friendly editor. It copies the look and feel of Pico, but is free software, and implements several features that Pico lacks, such as: opening multiple files, scrolling per line, undo/redo, syntax coloring, line numbering, and soft-wrapping overlong lines.

When giving a filename on the command line, the cursor can be put on a specific line by adding the line number with a plus sign (+) before the filename, and even in a specific column by adding it with a comma. (Negative numbers count from the end of the file or line.) The cursor can be put on the first or last occurrence of a specific string by specifying that string after +/ or +? before the filename. The string can be made case sensitive and/or caused to be interpreted as a regular expression by inserting **c** and/or **r** after the + sign. These search modes can be explicitly disabled by using the uppercase variant of those letters: **C** and/or **R**. When the string contains spaces, it needs to be enclosed in quotes. To give an example: to open a file at the first occurrence of the word "Foo", you would do:

nano +c/Foo file

As a special case: if instead of a filename a dash (–) is given, **nano** will read data from standard input.

EDITING

Entering text and moving around in a file is straightforward: typing the letters and using the normal cursor movement keys. Commands are entered by using the Control (^) and the Alt or Meta (M-) keys. Typing ^K deletes the current line and puts it in the cutbuffer. Consecutive ^Ks will put all deleted lines together in the cutbuffer. Any cursor movement or executing any other command will cause the next ^K to overwrite the cutbuffer. A^U will paste the current contents of the cutb uffer at the current cursor position.

When a more precise piece of text needs to be cut or copied, you can mark its start with ^G, move the cursor to its end (the marked text will be highlighted), and then use ^K to cut it, or M-6 to copy it to the cutbuffer. You can also save the marked text to a file with ^O, or spell check it with ^T^T.

On some terminals, text can be selected also by holding down Shift while using the arrow keys. Holding down the Ctrl or Alt key too will increase the stride. Any cursor movement without Shift being held will cancel such a selection.

The two lines at the bottom of the screen show some important commands; the built-in help (^G) lists all the available ones. The default key bindings can be changed via a *nanorc* file -- see **nanorc**(5).

NOTICE

Since version 4.0, **nano** by default:

- does not automatically hard-wrap lines that become overlong,
- includes the line below the title bar in the editing area,
- does linewise (smooth) scrolling.

If you want the old, Pico behavior back, you can use **--breaklonglines**, **--emptyline**, and **--jumppyscrolling** (or **-bej** for short).

OPTIONS**-A, --smarthome**

Make the Home key smarter. When Home is pressed anywhere but at the very beginning of non-whitespace characters on a line, the cursor will jump to that beginning (either forwards or backwards). If the cursor is already at that position, it will jump to the true beginning of the line.

-B, --backup

When saving a file, back up the previous version of it, using the current filename suffixed with a tilde (~).

-C directory, --backupdir=directory

Make and keep not just one backup file, but make and keep a uniquely numbered one every time a file is saved -- when backups are enabled (-B). The uniquely numbered files are stored in the specified *directory*.

-D, --boldtext

For the interface, use bold instead of reverse video. This will be overridden by setting the options **titlecolor**, **statuscolor**, **keycolor**, **functioncolor**, **numbercolor**, and/or **selectedcolor** in your *nanorc*(5). See **nanorc**(5).

-E, --tabstospaces

Convert each typed tab to spaces -- to the number of spaces that a tab at that position would take up.

-F, --multibuffer

Read a file into a new buffer by default.

-G, --locking

Use vim-style file locking when editing files.

-H, --historylog

Save the last hundred search strings and replacement strings and executed commands, so they can be easily reused in later sessions.

-I, --ignorercfiles

Don't look at the system's *nanorc* nor at the user's *nanorc*.

-J number, --guidestripe=number

Draw a vertical stripe at the given column, to help judge the width of the text. (The color of the stripe can be changed with **set stripecolor** in your *nanorc* file.)

-K, --rawsequences

Interpret escape sequences directly, instead of asking **ncurses** to translate them. (If you need this option to get some keys to work properly, it means that the terminfo terminal description that is used does not fully match the actual behavior of your terminal. This can happen when you ssh into a BSD machine, for example.) Using this option disables **nano**'s mouse support.

-L, --nonewlines

Don't automatically add a newline when a text does not end with one. (This can cause you to save non-POSIX text files.)

-M, --trimblanks

Snip trailing whitespace from the wrapped line when automatic hard-wrapping occurs or when text is justified.

-N, --noconvert

Disable automatic conversion of files from DOS/Mac format.

-O, --bookstyle

When justifying, treat any line that starts with whitespace as the beginning of a paragraph (unless auto-indenting is on).

-P, --positionlog

For the 200 most recent files, log the last position of the cursor, and place it at that position again upon reopening such a file.

-Q "regex", --quotestr="regex"

Set the regular expression for matching the quoting part of a line. The default value is `"^([\t]*([!#%:;>|{}]|//))+"`. (Note that `\t` stands for an actual T ab.) This makes it possible to rejustify blocks of quoted text when composing email, and to rewrap blocks of line comments when writing source code.

-R, --restricted

Restricted mode: don't read or write to any file not specified on the command line. This means: don't read or write history files; don't allow suspending; don't allow spell checking; don't allow a file to be appended to, prepended to, or saved under a different name if it already has one; and don't make backup files. Restricted mode can also be activated by invoking **nano** with any name beginning with 'r' (e.g. "rnano").

-S, --softwrap

Display over multiple screen rows lines that exceed the screen's width. (You can make this soft-wrapping occur at whitespace instead of rudely at the screen's edge, by using also **--atblanks**.) (The old short option, **-\$**, is deprecated.)

-T number, --tabsize=number

Set the size (width) of a tab to *number* columns. The value of *number* must be greater than 0. The default value is **8**.

-U, --quickblank

Make status-bar messages disappear after 1 keystroke instead of after 20. Note that option **-c** (**--constantshow**) overrides this. When option **--minibar** or **--zero** is in effect, **--quickblank** makes a message disappear after 0.8 seconds instead of after the default 1.5 seconds.

-V, --version

Show the current version number and exit.

-W, --wordbounds

Detect word boundaries differently by treating punctuation characters as part of a word.

-X "characters", --wordchars="characters"

Specify which other characters (besides the normal alphanumeric ones) should be considered as part of a word. When using this option, you probably want to omit **-W** (**--wordbounds**).

-Y name, --syntax=name

Specify the name of the syntax highlighting to use from among the ones defined in the *nanorc* files.

-Z, --zap

Let an unmodified Backspace or Delete erase the marked region (instead of a single character, and without affecting the cutbuffer).

-a, --atblanks

When doing soft line wrapping, wrap lines at whitespace instead of always at the edge of the screen.

-b, --breaklonglines

Automatically hard-wrap the current line when it becomes overlong. (This option is the opposite of **-w** (**--nowrap**) -- the last one given takes effect.)

-c, --constantshow

Constantly show the cursor position on the status bar. Note that this overrides option **-U** (**--quickblank**).

-d, --rebinddelete

Interpret the Delete and Backspace keys differently so that both Backspace and Delete work properly. You should only use this option when on your system either Backspace acts like Delete or Delete acts like Backspace.

-e, --emptyline

Do not use the line below the title bar, leaving it entirely blank.

-f file, --rcfile=file

Read only this *file* for setting nano's options, instead of reading both the system-wide and the user's nanorc files.

-g, --showcursor

Make the cursor visible in the file browser (putting it on the highlighted item) and in the help viewer. Useful for braille users and people with poor vision.

-h, --help

Show a summary of the available command-line options and exit.

-i, --autoindent

Automatically indent a newly created line to the same number of tabs and/or spaces as the previous line (or as the next line if the previous line is the beginning of a paragraph).

-j, --jumpyscrolling

Scroll the buffer contents per half-screen instead of per line.

-k, --cutfromcursor

Make the 'Cut Text' command (normally ^K) cut from the current cursor position to the end of the line, instead of cutting the entire line.

-l, --linenumbers

Display line numbers to the left of the text area. (Any line with an anchor additionally gets a mark in the margin.)

-m, --mouse

Enable mouse support, if available for your system. When enabled, mouse clicks can be used to place the cursor, set the mark (with a double click), and execute shortcuts. The mouse will work in the X Window System, and on the console when gpm is running. Text can still be selected through dragging by holding down the Shift key.

-n, --noread

Treat any name given on the command line as a new file. This allows **nano** to write to named pipes: it will start with a blank buffer, and will write to the pipe when the user saves the "file". This way **nano** can be used as an editor in combination with for instance **gpg** without having to write sensitive data to disk first.

-o directory, --operatingdir=directory

Set the operating directory. This makes **nano** set up something similar to a chroot.

-p, --preserve

Preserve the XON and XOFF sequences (^Q and ^S) so they will be caught by the terminal.

-q, --indicator

Display a "scrollbar" on the righthand side of the edit window. It shows the position of the viewport in the buffer and how much of the buffer is covered by the viewport.

-r number, --fill=number

Set the target width for justifying and automatic hard-wrapping at this *number* of columns. If the value is 0 or less, wrapping will occur at the width of the screen minus *number* columns, allowing the wrap point to vary along with the width of the screen if the screen is resized. The default value is -8.

-s "program [argument ...]", --spell="program [argument ...]"

Use this command to perform spell checking and correcting, instead of using the built-in corrector that calls **hunspell(1)** or **spell(1)**.

-t, --saveonexit

Save a changed buffer without prompting (when exiting with **^X**). (The old form of the long option, **--tempfile**, is deprecated.)

-u, --unix

Save a file by default in Unix format. This overrides nano's default behavior of saving a file in the format that it had. (This option has no effect when you also use **--noconvert**.)

-v, --view

Just view the file and disallow editing: read-only mode. This mode allows the user to open also other files for viewing, unless **--restricted** is given too.

-w, --nowrap

Do not automatically hard-wrap the current line when it becomes overlong. This is the default. (This option is the opposite of **-b (--breaklonglines)** -- the last one given takes effect.)

-x, --nohelp

Don't show the two help lines at the bottom of the screen.

-y, --afterends

Make Ctrl+Right and Ctrl+Delete stop at word ends instead of beginnings.

-z, --suspendable

Obsolete option; ignored. Suspension is enabled by default, reachable via **^T^Z**. (If you want a plain **^Z** to suspend nano, add **bind ^Z suspend main** to your nanorc.)

-%, --stateflags

Use the top-right corner of the screen for showing some state flags: **I** when auto-indenting, **M** when the mark is on, **L** when hard-wrapping (breaking long lines), **R** when recording a macro, and **S** when soft-wrapping. When the buffer is modified, a star (*) is shown after the filename in the center of the title bar.

-_, --minibar

Suppress the title bar and instead show information about the current buffer at the bottom of the screen, in the space for the status bar. In this "minibar" the filename is shown on the left, followed by an asterisk if the buffer has been modified. On the right are displayed the current line and column number, the code of the character under the cursor (in Unicode format: U+xxxx), the same flags as are shown by **--stateflags**, and a percentage that expresses how far the cursor is into the file (linewise). When a file is loaded or saved, and also when switching between buffers, the number of lines in the buffer is displayed after the filename. This number is cleared upon the next keystroke, or replaced with an [i/n] counter when multiple buffers are open. The line plus column numbers and the character code are displayed only when **--constantshow** is used, and can be toggled on and off with **M-C**. The state flags are displayed only when **--stateflags** is used.

-0, --zero

Hide all elements of the interface (title bar, status bar, and help lines) and use all rows of the terminal for showing the contents of the buffer. The status bar appears only when there is a significant message, and disappears after 1.5 seconds or upon the next keystroke. With **M-Z** the title bar plus status bar can be toggled. With **M-X** the help lines.

-!, --magic

When neither the file's name nor its first line give a clue, try using libmagic to determine the applicable syntax.

TOGGLERS

Several of the above options can be switched on and off also while **nano** is running. For example, **M-L** toggles the hard-wrapping of long lines, **M-S** toggles soft-wrapping, **M-N** toggles line numbers, **M-M**

toggles the mouse, **M-I** auto-indentation, and **M-X** the help lines. See at the end of the **^G** help text for a complete list.

The **M-X** toggle is special: it works in all menus except the help viewer and the linter. All other toggles work in the main menu only.

FILES

When **--rcfile** is given, **nano** will read just the specified file for setting its options and syntaxes and key bindings. Without that option, **nano** will read two configuration files: first the system's *nanorc* (if it exists), and then the user's *nanorc* (if it exists), either *~.nanorc* or *\$XDG_CONFIG_HOME/nano/nanorc* or *~.config/nano/nanorc*, whichever is encountered first. See **nanorc(5)** for more information on the possible contents of those files.

See */usr/share/nano/* and */usr/share/nano/extral/* for available syntax-coloring definitions.

NOTES

If no alternative spell checker command is specified on the command line nor in one of the *nanorc* files, **nano** will check the **SPELL** environment variable for one.

In some cases **nano** will try to dump the buffer into an emergency file. This will happen mainly if **nano** receives a SIGHUP or SIGTERM or runs out of memory. It will write the buffer into a file named *nano.save* if the buffer didn't have a name already, or will add a ".save" suffix to the current filename. If an emergency file with that name already exists in the current directory, it will add ".save" plus a number (e.g. ".save.1") to the current filename in order to make it unique. In multibuffer mode, **nano** will write all the open buffers to their respective emergency files.

BUGS

The recording and playback of keyboard macros works correctly only on a terminal emulator, not on a Linux console (VT), because the latter does not by default distinguish modified from unmodified arrow keys.

Please report any other bugs that you encounter via:
<https://savannah.gnu.org/bugs/?group=nano>.

When nano crashes, it will save any modified buffers to emergency .save files. If you are able to reproduce the crash and you want to get a backtrace, define the environment variable **NANO_NOCATCH**.

HOMEPAGE

<https://nano-editor.org/>

SEE ALSO

nanorc(5)

/usr/share/doc/nano/ (or equivalent on your system)

NAME

nanorc – GNU nano’s configuration file

DESCRIPTION

The *nanorc* files contain the default settings for **nano**, a small and friendly editor. During startup, if **--rcfile** is not given, **nano** will read two files: first the system-wide settings, from */etc/nanorc* (the exact path might be different on your system), and then the user-specific settings, either from *~/.nanorc* or from *\$XDG_CONFIG_HOME/nano/nanorc* or from *~/.config/nano/nanorc*, whichever is encountered first. If **--rcfile** is given, **nano** will read just the specified settings file.

NOTICE

Since version 4.0, **nano** by default:

- does not automatically hard-wrap lines that become overlong,
- includes the line below the title bar in the editing area,
- does linewise (smooth) scrolling.

To get the old, Pico behavior back, you can use **set breaklonglines**, **set emptyline**, and **set jumpyscrolling**.

OPTIONS

The configuration file accepts a series of **set** and **unset** commands, which can be used to configure nano on startup without using command-line options. Additionally, there are some commands to define syntax highlighting and to rebind keys -- see the two separate sections on those. **nano** reads one command per line. All commands and keywords should be written in lowercase.

Options in *nanorc* files take precedence over nano’s defaults, and command-line options override *nanorc* settings. Also, options that do not take an argument are unset by default. So using the **unset** command is only needed when wanting to override a setting of the system’s *nanorc* file in your own *nanorc*. Options that take an argument cannot be unset.

Quotes inside the *characters* parameters below should not be escaped. The last double quote on the line will be seen as the closing quote.

The supported commands and arguments are:

set afterends

Make Ctrl+Right and Ctrl+Delete stop at word ends instead of beginnings.

set allow_insecure_backup

When backing up files, allow the backup to succeed even if its permissions can’t be (re)set due to special OS considerations. You should NOT enable this option unless you are sure you need it.

set atblanks

When soft line wrapping is enabled, make it wrap lines at blank characters (tabs and spaces) instead of always at the edge of the screen.

set autoindent

Automatically indent a newly created line to the same number of tabs and/or spaces as the previous line (or as the next line if the previous line is the beginning of a paragraph).

set backup

When saving a file, create a backup file by adding a tilde (~) to the file’s name.

set backupdir *directory*

Make and keep not just one backup file, but make and keep a uniquely numbered one every time a file is saved -- when backups are enabled with **set backup** or **--backup** or **-B**. The uniquely numbered files

are stored in the specified *directory*.

set boldtext

Use bold instead of reverse video for the title bar, status bar, key combos, function tags, line numbers, and selected text. This can be overridden by setting the options **titlecolor**, **statuscolor**, **keycolor**, **functioncolor**, **numbercolor**, and **selectedcolor**.

set bookstyle

When justifying, treat any line that starts with whitespace as the beginning of a paragraph (unless auto-indenting is on).

set brackets "characters"

Set the characters treated as closing brackets when justifying paragraphs. This may not include blank characters. Only closing punctuation (seeset **punct**), optionally followed by the specified closing brackets, can end sentences. The default value is """)>}".

set breaklonglines

Automatically hard-wrap the current line when it becomes overlong.

set casesensitive

Do case-sensitive searches by default.

set constantshow

Constantly display the cursor position in the status bar. This overrides the option **quickblank**.

set cutfromcursor

Use cut-from-cursor-to-end-of-line by default, instead of cutting the whole line.

set emptyline

Do not use the line below the title bar, leaving it entirely blank.

set errorcolor [bold,][italic,]fgcolor,bgcolor

Use this color combination for the status bar when an error message is displayed. The default value is **bold,white,red**. Seeset **titlecolor** for valid color names.

set fill number

Set the target width for justifying and automatic hard-wrapping at this *number* of columns. If the value is 0 or less, wrapping will occur at the width of the screen minus *number* columns, allowing the wrap point to vary along with the width of the screen if the screen is resized. The default value is **-8**.

set functioncolor [bold,][italic,]fgcolor,bgcolor

Use this color combination for the concise function descriptions in the two help lines at the bottom of the screen. See set **titlecolor** for more details.

set guidestripe number

Draw a vertical stripe at the given column, to help judge the width of the text. (The color of the stripe can be changed with set **stripecolor**.)

set historylog

Save the last hundred search strings and replacement strings and executed commands, so they can be easily reused in later sessions.

set indicator

Display a "scrollbar" on the righthand side of the edit window. It shows the position of the viewport in the buffer and how much of the buffer is covered by the viewport.

set jumpyscrolling

Scroll the buffer contents per half-screen instead of per line.

set keycolor [bold,][italic,]fgcolor,bgcolor

Use this color combination for the shortcut key combos in the two help lines at the bottom of the screen. See set **titlecolor** for more details.

set linenumbers

Display line numbers to the left of the text area. (Any line with an anchor additionally gets a mark in the margin.)

set locking

Enable vim-style lock-files for when editing files.

set magic

When neither the file's name nor its first line give a clue, try using libmagic to determine the applicable syntax. (Calling libmagic can be relatively time consuming. It is therefore not done by default.)

set matchbrackets "*characters*"

Specify the opening and closing brackets that can be found by bracket searches. This may not include blank characters. The opening set must come before the closing set, and the two sets must be in the same order. The default value is "<[{}]>".

set minibar

Suppress the title bar and instead show information about the current buffer at the bottom of the screen, in the space for the status bar. In this "minibar" the filename is shown on the left, followed by an asterisk if the buffer has been modified. On the right are displayed the current line and column number, the code of the character under the cursor (in Unicode format: U+xxxx), the same flags as are shown by **set stateflags**, and a percentage that expresses how far the cursor is into the file (linewise). When a file is loaded or saved, and also when switching between buffers, the number of lines in the buffer is displayed after the filename. This number is cleared upon the next keystroke, or replaced with an [i/n] counter when multiple buffers are open. The line plus column numbers and the character code are displayed only when **set constantshow** is used, and can be toggled on and off with **M-C**. The state flags are displayed only when **set stateflags** is used.

set minicolor [bold**,][**italic**,]*fgcolor,bgcolor***

Use this color combination for the minibar. (When this option is not specified, the colors of the title bar are used.) See **set titlecolor** for more details.

set mouse

Enable mouse support, if available for your system. When enabled, mouse clicks can be used to place the cursor, set the mark (with a double click), and execute shortcuts. The mouse will work in the X Window System, and on the console when gpm is running. Text can still be selected through dragging by holding down the Shift key.

set multibuffer

When reading in a file with **^R**, insert it into a new buffer by default.

set noconvert

Don't convert files from DOS/Mac format.

set nohelp

Don't display the two help lines at the bottom of the screen.

set nonewlines

Don't automatically add a newline when a text does not end with one. (This can cause you to save non-POSIX text files.)

set nowrap

Deprecated option since it has become the default setting. When needed, use **unset breaklonglines** instead.

set numbercolor [bold**,][**italic**,]*fgcolor,bgcolor***

Use this color combination for line numbers. See **set titlecolor** for more details.

set operatingdir *directory*

nano will only read and write files inside *directory* and its subdirectories. Also, the current directory is changed to here, so files are inserted from this directory. By default, the operating directory feature is turned off.

set positionlog

Save the cursor position of files between editing sessions. The cursor position is remembered for the 200 most-recently edited files.

set preserve

Preserve the XON and XOFF keys (^Q and ^S).

set promptcolor [bold,][italic,]*fgcolor,bgcolor*

Use this color combination for the prompt bar. (When this option is not specified, the colors of the title bar are used.) See **set titlecolor** for more details.

set punct "characters"

Set the characters treated as closing punctuation when justifying paragraphs. This may not include blank characters. Only the specified closing punctuation, optionally followed by closing brackets (see **brackets**), can end sentences. The default value is ".?".

set quickblank

Make status-bar messages disappear after 1 keystroke instead of after 20. Note that option **constantshow** overrides this. When option **minibar** or **zero** is in effect, **quickblank** makes a message disappear after 0.8 seconds instead of after the default 1.5 seconds.

set quotestr "regex"

Set the regular expression for matching the quoting part of a line. The default value is `^([\t]*([!#%:;:>|])]+` . (Note that \t stands for an actual Tab character.) This makes it possible to rejustify blocks of quoted text when composing email, and to rewrap blocks of line comments when writing source code.

set rawsequences

Interpret escape sequences directly, instead of asking **ncurses** to translate them. (If you need this option to get some keys to work properly, it means that the terminfo terminal description that is used does not fully match the actual behavior of your terminal. This can happen when you ssh into a BSD machine, for example.) Using this option disables **nano**'s mouse support.

set rebinddelete

Interpret the Delete and Backspace keys differently so that both Backspace and Delete work properly. You should only use this option when on your system either Backspace acts like Delete or Delete acts like Backspace.

set regexp

Do regular-expression searches by default. Regular expressions in **nano** are of the extended type (ERE).

set saveonexit

Save a changed buffer automatically on exit (^X); don't prompt. (The old form of this option, **set tempfile**, is deprecated.)

set scrollercolor *fgcolor,bgcolor*

Use this color combination for the indicator alias "scrollbar". (On terminal emulators that link to a libvte older than version 0.55, using a background color here does not work correctly.) See **set titlecolor** for more details.

set selectedcolor [bold,][italic,]*fgcolor,bgcolor*

Use this color combination for selected text. See **set titlecolor** for more details.

set showcursor

Put the cursor on the highlighted item in the file browser, and show the cursor in the help viewer, to aid braille users and people with poor vision.

set smarthome

Make the Home key smarter. When Home is pressed anywhere but at the very beginning of non-white-space characters on a line, the cursor will jump to that beginning (either forwards or backwards). If the cursor is already at that position, it will jump to the true beginning of the line.

set softwrap

Display lines that exceed the screen's width over multiple screen lines. (You can make this soft-wrappping occur at whitespace instead of rudely at the screen's edge, by using also **set atblanks**.)

set speller "program [argument ...]"

Use the given *program* to do spell checking and correcting, instead of using the built-in corrector that calls **hunspell(1)** or **spell(1)**.

set spotlightcolor [bold,][italic,]*fgcolor*,*bgcolor*

Use this color combination for highlighting a search match. The default value is **black,lightyellow**.

See **set titlecolor** for valid color names.

set stateflags

Use the top-right corner of the screen for showing some state flags: **I** when auto-indenting, **M** when the mark is on, **L** when hard-wrapping (breaking long lines), **R** when recording a macro, and **S** when soft-wrapping. When the buffer is modified, a star (*) is shown after the filename in the center of the title bar.

set statuscolor [bold,][italic,]*fgcolor*,*bgcolor*

Use this color combination for the status bar. See **set titlecolor** for more details.

set stripecolor [bold,][italic,]*fgcolor*,*bgcolor*

Use this color combination for the vertical guiding stripe. See **set titlecolor** for more details.

set suspendable

Obsolete option; ignored. Suspension is enabled by default, reachable via **^T^Z**. (If you want a plain **Z** to suspend nano, add **bind 'Z suspend main** to your nanorc.)

set tabsize *number*

Use a tab size of *number* columns. The value of *number* must be greater than 0. The default value is **8**.

set tabstospaces

Convert each typed tab to spaces -- to the number of spaces that a tab at that position would take up.

set titlecolor [bold,][italic,]*fgcolor*,*bgcolor*

Use this color combination for the title bar. Valid names for the foreground and background colors are: **red**, **green**, **blue**, **magenta**, **yellow**, **cyan**, **white**, and **black**. Each of these eight names may be pre-fixed with the word **light** to get a brighter version of that color. The word **grey** or **gray** may be used as a synonym for **lightblack**. On terminal emulators that can do at least 256 colors, other valid (but unprefixable) color names are: **pink**, **purple**, **mauve**, **lagoon**, **mint**, **lime**, **peach**, **orange**, **latte**, **rosy**, **beet**, **plum**, **sea**, **sky**, **slate**, **teal**, **sage**, **brown**, **ocher**, **sand**, **tawny**, **brick**, **crimson**, and **normal** -- where **normal** means the default foreground or background color. On such emulators, the color may also be specified as a three-digit hexadecimal number prefixed with #, with the digits representing the amounts of red, green, and blue, respectively. This tellsnano to select from the available palette the color that approximates the given values.

Either "*fgcolor*" or ",*bgcolor*" may be left out, and the pair may be preceded by **bold** and/or **italic** (separated by commas) to get a bold and/or slanting typeface, if your terminal can do those.

set trimblanks

Remove trailing whitespace from wrapped lines when automatic hard-wrapping occurs or when text is justified.

set unix

Save a file by default in Unix format. This overrides nano's default behavior of saving a file in the format that it had. (This option has no effect when you also use **set noconvert**.)

set whitespace "*characters*"

Set the two characters used to indicate the presence of tabs and spaces. They must be single-column characters. The default pair for a UTF-8 locale is "».", and for other locales ">.".

set wordbounds

Detect word boundaries differently by treating punctuation characters as parts of words.

set wordchars "characters"

Specify which other characters (besides the normal alphanumeric ones) should be considered as parts of words. When using this option, you probably want to unset **wordbounds**.

set zap

Let an unmodified Backspace or Delete erase the marked region (instead of a single character, and without affecting the cutbuffer).

set zero

Hide all elements of the interface (title bar, status bar, and help lines) and use all rows of the terminal for showing the contents of the buffer. The status bar appears only when there is a significant message, and disappears after 1.5 seconds or upon the next keystroke. With **M-Z** the title bar plus status bar can be toggled. With **M-X** the help lines.

SYNTAX HIGHLIGHTING

Coloring the different syntactic elements of a file is done via regular expressions (see the **color** command below). This is inherently imperfect, because regular expressions are not powerful enough to fully parse a file. Nevertheless, regular expressions can do a lot and are easy to make, so they are a good fit for a small editor like **nano**.

All regular expressions in **nano** are POSIX extended regular expressions. This means that., ?, *, +, ^, \$, and several other characters are special. The period . matches any single character, ? means the preceding item is optional, * means the preceding item may be matched zero or more times, + means the preceding item must be matched one or more times, ^ matches the beginning of a line, and \$ the end, \< matches the start of a word, and \> the end, and \s matches a blank. It also means that lookahead and lookbehind are not possible. A complete explanation can be found in the manual page of GNU grep: **man grep**.

For each kind of file a separate syntax can be defined via the following commands:

syntax name ["fileregex" ...]

Start the definition of a syntax with this *name*. All subsequent **color** and other such commands will be added to this syntax, until a new **syntax** command is encountered.

When **nano** is run, this syntax will be automatically activated if the current filename matches the extended regular expression *fileregex*. Or the syntax can be explicitly activated by using the **-Y** or **--syntax** command-line option followed by the *name*.

The syntax **default** is special: it takes no *fileregex*, and applies to files that don't match any syntax's regexes. The **syntaxnone** is reserved; specifying it on the command line is the same as not having a syntax at all.

header "regex" ...

If from all defined syntaxes no *fileregex* matched, then compare this *regex* (or regexes) against the first line of the current file, to determine whether this syntax should be used for it.

magic "regex" ...

If no *fileregex* matched and no **header** regex matched either, then compare this *regex* (or regexes) against the result of querying the **magic** database about the current file, to determine whether this syntax should be used for it. (This functionality only works when **libmagic** is installed on the system and will be silently ignored otherwise.)

formatter program [argument ...]

Run the given *program* on the full contents of the current buffer. (The current buffer is written out to a temporary file, the program is run on it, and then the temporary file is read back in, replacing the contents of the buffer.)

linter *program [argument ...]*

Use the given *program* to run a syntax check on the current buffer.

comment "string"

Use the given *string* for commenting and uncommenting lines. If the string contains a vertical bar or pipe character (|), this designates bracket-style comments; for example, "/*|*/" for CSS files. The characters before the pipe are prepended to the line and the characters after the pipe are appended at the end of the line. If no pipe character is present, the full string is prepended; for example, "#" for Python files. If empty double quotes are specified, the comment/uncomment function is disabled; for example, "" for JSON. The default value is "#".

tabgives "string"

Make the <Tab> key produce the given *string*. Useful for languages like Python that want to see only spaces for indentation. This overrides the setting of the **tabstospaces** option.

color [**bold**,][**italic**,]*fgcolor,bgcolor "regex"* ...

Paint all pieces of text that match the extended regular expression *regex* with the given foreground and background colors, at least one of which must be specified. Valid color names are: **red**, **green**, **blue**, **magenta**, **yellow**, **cyan**, **white**, and **black**. Each of these eight names may be prefixed with the word **light** to get a brighter version of that color. The word **grey** or **gray** may be used as a synonym for **lightblack**. On terminal emulators that can do at least 256 colors, other valid (but unprefixable) color names are: **pink**, **purple**, **mauve**, **lagoon**, **mint**, **lime**, **peach**, **orange**, **latte**, **rosy**, **beet**, **plum**, **sea**, **sky**, **slate**, **teal**, **sage**, **brown**, **ocher**, **sand**, **tawny**, **brick**, **crimson**, and **normal** -- where **normal** means the default foreground or background color. On such emulators, the color may also be specified as a three-digit hexadecimal number prefixed with #, with the digits representing the amounts of red, green, and blue, respectively. This tells nano to select from the available palette the color that approximates the given values.

The color pair may be preceded by **bold** and/or **italic** (separated by commas) to get a bold and/or slanting typeface, if your terminal can do those.

All coloring commands are applied in the order in which they are specified, which means that later commands can recolor stuff that was colored earlier.

icolor [**bold**,][**italic**,]*fgcolor,bgcolor "regex"* ...

Same as above, except that the matching is case insensitive.

color [**bold**,][**italic**,]*fgcolor,bgcolor start="fromrx" end="torx"*

Paint all pieces of text whose start matches extended regular expression *fromrx* and whose end matches extended regular expression *torx* with the given foreground and background colors, at least one of which must be specified. This means that, after an initial instance of *fromrx*, all text until the first instance of *torx* will be colored. This allows syntax highlighting to span multiple lines.

icolor [**bold**,][**italic**,]*fgcolor,bgcolor start="fromrx" end="torx"*

Same as above, except that the matching is case insensitive.

include "*syntaxfile*"

Read in self-contained color syntaxes from *syntaxfile*. Note that *syntaxfile* may contain only the above commands, from **syntax** to **icolor**.

extendsyntax *name command argument* ...

Extend the syntax previously defined as *name* with another *command*. This allows adding a new **color**, **icolor**, **header**, **magic**, **formatter**, **linter**, **comment**, or **tabgives** command to an already defined syntax -- useful when you want to slightly improve a syntax defined in one of the system-installed files (which normally are not writable).

REBINDING KEYS

Key bindings can be changed via the following three commands:

bind *key function menu*

Rebinds the given *key* to the given *function* in the given *menu* (or in all menus where the function exists when **all** is used).

bind *key "string" menu*

Makes the given *key* produce the given *string* in the given *menu* (or in all menus where the key exists when **all** is used). The *string* can consist of text or commands or a mix of them. (To enter a command into the *string*, precede its keystroke with **M-V**.)

unbind *key menu*

Unbinds the given *key* from the given *menu* (or from all menus where the key exists when **all** is used).

The format of *key* should be one of:

^X where *X* is a Latin letter, or one of several ASCII characters (@,], \, ^, _), or the word "Space".
Example: **^C**.

M-X where *X* is any ASCII character except [, or the word "Space". Example: **M-8**.

Sh-M-X

where *X* is a Latin letter. Example: **Sh-M-U**. By default, each Meta+letter keystroke does the same as the corresponding Shift+Meta+letter. But when any Shift+Meta bind is made, that will no longer be the case, for all letters.

FN where *N* is a numeric value from 1 to 24. Example: **F10**. (Often, **F13** to **F24** can be typed as **F1** to **F12** with Shift.)

Ins or **Del**.

Rebinding **^M** (Enter) or **^T** (Tab) is probably not a good idea. Rebinding **^L** (Esc) is not possible, because its keycode is the starter byte of Meta keystrokes and escape sequences. Rebinding any of the dedicated cursor-moving keys (the arrows, Home, End, PageUp and PageDown) is not possible. On some terminals it's not possible to rebind **^H** (unless **--raw** is used) because its keycode is identical to that of the Backspace key.

Valid *function* names to be bound are:

help

Invokes the help viewer.

cancel

Cancels the current command.

exit

Exits from the program (or from the help viewer or file browser).

writeout

Writes the current buffer to disk, asking for a name.

savefile

Writes the current file to disk without prompting.

insert

Inserts a file into the current buffer (at the current cursor position), or into a new buffer when option **multibuffer** is set.

whereis

Starts a forward search for text in the current buffer -- or for filenames matching a string in the current list in the file browser.

wherewas

Starts a backward search for text in the current buffer -- or for filenames matching a string in the current list in the file browser.

findprevious

Searches the next occurrence in the backward direction.

findnext

Searches the next occurrence in the forward direction.

replace

Interactively replaces text within the current buffer.

cut

Cuts and stores the current line (or the marked region).

copy

Copies the current line (or the marked region) without deleting it.

paste

Pastes the currently stored text into the current buffer at the current cursor position.

zap

Throws away the current line (or the marked region). (This function is bound by default to <Meta+Delete>.)

chopwordleft

Deletes from the cursor position to the beginning of the preceding word. (This function is bound by default to <Shift+Ctrl+Delete>. If your terminal produces **H** for <Ctrl+Backspace>, you can make <Ctrl+Backspace> delete the word to the left of the cursor by rebinding **H** to this function.)

chopwordright

Deletes from the cursor position to the beginning of the next word. (This function is bound by default to <Ctrl+Delete>.)

cutrestoffile

Cuts all text from the cursor position till the end of the buffer.

mark

Sets the mark at the current position, to start selecting text. Or, when it is set, unsets the mark.

location

Reports the current position of the cursor in the buffer: the line, column, and character positions. (The old name of this function, 'curpos', is deprecated.)

wordcount

Counts and reports on the status bar the number of lines, words, and characters in the current buffer (or in the marked region).

execute

Prompts for a program to execute. The program's output will be inserted into the current buffer (or into a new buffer when **M-F** is toggled).

speller

Invokes a spell-checking program, either the default **hunspell(1)** or GNU **spell(1)**, or the one defined by **--speller** or **set speller**.

formatter

Invokes a full-buffer-processing program (if the active syntax defines one).

linter

Invokes a syntax-checking program (if the active syntax defines one).

justify

Justifies the current paragraph (or the marked region). A paragraph is a group of contiguous lines that, apart from possibly the first line, all have the same indentation. The beginning of a paragraph is detected by either this lone line with a differing indentation or by a preceding blank line.

fulljustify

Justifies the entire current buffer (or the marked region).

indent

Indents (shifts to the right) the current line or the marked lines.

unindent

Unindents (shifts to the left) the current line or the marked lines.

comment

Comments or uncomments the current line or the marked lines, using the comment style specified in the active syntax.

complete

Completes (when possible) the fragment before the cursor to a full word found elsewhere in the current buffer.

left

Goes left one position (in the editor or browser).

right

Goes right one position (in the editor or browser).

up

Goes one line up (in the editor or browser).

down

Goes one line down (in the editor or browser).

scrollup

Scrolls the viewport up one row (meaning that the text slides down) while keeping the cursor in the same text position, if possible. (This function is bound by default to <Alt+Up>. If <Alt+Up> does nothing on your Linux console, see the FAQ: <https://nano-editor.org/dist/latest/faq.html#4.1>.)

scrolldown

Scrolls the viewport down one row (meaning that the text slides up) while keeping the cursor in the same text position, if possible. (This function is bound by default to <Alt+Down>.)

center

Scrolls the line with the cursor to the middle of the screen.

prevword

Moves the cursor to the beginning of the previous word.

nextword

Moves the cursor to the beginning of the next word.

home

Moves the cursor to the beginning of the current line.

end

Moves the cursor to the end of the current line.

beginpara

Moves the cursor to the beginning of the current paragraph.

endpara

Moves the cursor to the end of the current paragraph.

prevblock

Moves the cursor to the beginning of the current or preceding block of text. (Blocks are separated by one or more blank lines.)

nextblock

Moves the cursor to the beginning of the next block of text.

pageup

Goes up one screenful.

pagedown

Goes down one screenful.

firstline

Goes to the first line of the file.

lastline

Goes to the last line of the file.

gotoline

Goes to a specific line (and column if specified). Negative numbers count from the end of the file (and end of the line).

findbracket

Moves the cursor to the bracket (or brace or parenthesis, etc.) that matches (pairs) with the one under the cursor. Seeset **matchbrack ets**.

anchor

Places an anchor at the current line, or removes it when already present. (An anchor is visible when line numbers are activated.)

prevanchor

Goes to the first anchor before the current line.

nextanchor

Goes to the first anchor after the current line.

prevbuf

Switches to editing/viewing the previous buffer when multiple buffers are open.

nextbuf

Switches to editing/viewing the next buffer when multiple buffers are open.

verbatim

Inserts the next keystroke verbatim into the file.

tab

Inserts a tab at the current cursor location.

enter

Inserts a new line below the current one.

delete

Deletes the character under the cursor.

backspace

Deletes the character before the cursor.

recordmacro

Starts the recording of keystrokes -- the keystrokes are stored as a macro. When already recording, the recording is stopped.

runmacro

Replays the keystrokes of the last recorded macro.

undo

Undoes the last performed text action (add text, delete text, etc).

redo

Redoes the last undone action (i.e., it undoes an undo).

refresh

Refreshes the screen.

suspend

Suspends the editor and returns control to the shell (until you tell the process to resume execution with **fg**).

casesens

Toggles whether searching/replacing ignores or respects the case of the given characters.

regexp

Toggles whether searching/replacing uses literal strings or regular expressions.

backwards

Toggles whether searching/replacing goes forward or backward.

older

Retrieves the previous (earlier) entry at a prompt.

newer

Retrieves the next (later) entry at a prompt.

flipreplace

Toggles between searching for something and replacing something.

flipgoto

Toggles between searching for text and targeting a line number.

flipexecute

Toggles between inserting a file and executing a command.

flippipe

When executing a command, toggles whether the current buffer (or marked region) is piped to the command.

flipnewbuffer

Toggles between inserting into the current buffer and into a new empty buffer.

flipconvert

When reading in a file, toggles between converting and not converting it from DOS/Mac format. Converting is the default.

dosformat

When writing a file, switches to writing a DOS format (CR/LF).

macformat

When writing a file, switches to writing a Mac format.

append

When writing a file, appends to the end instead of overwriting.

prepend

When writing a file, 'prepends' (writes at the beginning) instead of overwriting.

backup

When writing a file, creates a backup of the current file.

discardbuffer

When about to write a file, discard the current buffer without saving. (This function is bound by default only when option **--saveonexit** is in effect.)

browser

Starts the file browser (in the Read File and Write Out menus), allowing to select a file from a list.

gotodir

Goes to a directory to be specified, allowing to browse anywhere in the filesystem.

firstfile

Goes to the first file in the list when using the file browser.

lastfile

Goes to the last file in the list when using the file browser.

nohelp

Toggles the presence of the two-line list of key bindings at the bottom of the screen. (This toggle is special: it is available in all menus except the help viewer and the linter. All further toggles are available in the main menu only.)

zero

Toggles the presence of title bar and status bar.

constantshow

Toggles the constant display of the current line, column, and character positions.

softwrap

Toggles the displaying of overlong lines on multiple screen lines.

linenumbers

Toggles the display of line numbers in front of the text.

whitespacedisplay

Toggles the showing of whitespace.

nosyntax

Toggles syntax highlighting.

smarthome

Toggles the smartness of the Home key.

autoindent

Toggles whether a newly created line will contain the same amount of leading whitespace as the preceding line -- or as the next line if the preceding line is the beginning of a paragraph.

cutfromcursor

Toggles whether cutting text will cut the whole line or just from the current cursor position to the end of the line.

breaklonglines

Toggles whether long lines will be hard-wrapped to the next line. (The old name of this function, 'nowrap', is deprecated.)

tabstospaces

Toggles whether typed tabs will be converted to spaces.

mouse

Toggles mouse support.

Valid *menu* sections are:

main

The main editor window where text is entered and edited.

help

The help-viewer menu.

search

The search menu (AKA whereis).

replace

The 'search to replace' menu.

replacewith

The 'replace with' menu, which comes up after 'search to replace'.

yesno

The 'yesno' menu, where the Yes/No/All/Cancel question is asked.

gotoline

The 'goto line (and column)' menu.

writeout

The 'write file' menu.

insert

The 'insert file' menu.

browser

The 'file browser' menu, for selecting a file to be opened or inserted or written to.

whereisfile

The 'search for a file' menu in the file browser.

gotodir

The 'go to directory' menu in the file browser.

execute

The menu for inserting the output from an external command, or for filtering the buffer (or the marked region) through an external command, or for executing one of several tools. (The old form of this menu name, 'extcmd', is deprecated.)

spell

The menu of the integrated spell checker where the user can edit a misspelled word.

linter

The linter menu, which allows jumping through the linting messages.

all

A special name that encompasses all menus. For **bind** it means all menus where the specified *function* exists; for **unbind** it means all menus where the specified *key* exists.

FILES

/etc/nanorc

System-wide configuration file.

/.nanorc or *\$XDG_CONFIG_HOME/nano/nanorc* or *~/.config/nano/nanorc*

Per-user configuration file.

*/usr/share/nano/**

Syntax definitions for the syntax coloring of common file types (and for less common file types in the *extra/* subdirectory).

SEE ALSO

nano(1)

<https://nano-editor.org/cheatsheet.html>

An overview of the default key bindings.

NAME

netstat – Print network connections, routing tables, interface statistics, masquerade connections, and multi-cast memberships

SYNOPSIS

```
netstat [address_family_options] [--tcp|-t] [--udp|-u] [--udplite|-U] [--sctp|-S] [--raw|-w]
[--l2cap|-2] [--rfcomm|-f] [--listening|-l] [--all|-a] [--numeric|-n] [--numeric-hosts] [--numeric-ports]
[--numeric-users] [--symbolic|-N] [--extend|-e|--extend|-e] [--timers|-o] [--program|-p]
[--verbose|-v] [--continuous|-c] [--wide|-W]

netstat {--route|-r} [address_family_options] [--extend|-e|--extend|-e] [--verbose|-v] [--numeric|-n]
[--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]

netstat {--interfaces|-i} [--all|-a] [--extend|-e|--extend|-e] [--verbose|-v] [--program|-p]
[--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users] [--continuous|-c]

netstat {--groups|-g} [--numeric|-n] [--numeric-hosts] [--numeric-ports] [--numeric-users]
[--continuous|-c]

netstat {--masquerade|-M} [--extend|-e] [--numeric|-n] [--numeric-hosts] [--numeric-ports]
[--numeric-users] [--continuous|-c]

netstat {--statistics|-s} [--tcp|-t] [--udp|-u] [--udplite|-U] [--sctp|-S] [--raw|-w]

netstat {--version|-V}

netstat {--help|-h}

address_family_options:
[-4|--inet] [-6|--inet6] [--protocol={inet,inet6,unix,ipx,ax25,netrom,ddp,bluetooth,...}] 
[--unix|-x] [--inet|--ip|--tcpip] [--ax25] [--x25] [--rose] [--ash] [--bluetooth] [--ipx]
[--netrom] [--ddp|--appletalk] [--econet|--ec]
```

NOTES

This program is mostly obsolete. Replacement for **netstat** is **ss**. Replacement for **netstat -r** is **ip route**. Replacement for **netstat -i** is **ip -s link**. Replacement for **netstat -g** is **ip maddr**.

DESCRIPTION

Netstat prints information about the Linux networking subsystem. The type of information printed is controlled by the first argument, as follows:

(none)

By default, **netstat** displays a list of open sockets. If you don't specify any address families, then the active sockets of all configured address families will be printed.

--route, -r

Display the kernel routing tables. See the description in **route(8)** for details. **netstat -r** and **r oute** produce the same output.

--groups, -g

Display multicast group membership information for IPv4 and IPv6.

--interfaces, -i

Display a table of all network interfaces.

--masquerade, -M

Display a list of masqueraded connections.

--statistics, -s

Display summary statistics for each protocol.

OPTIONS

--verbose, -v

Tell the user what is going on by being verbose. Especially print some useful information about unconfigured address families.

--wide, -W

Do not truncate IP addresses by using output as wide as needed. This is optional for now to not break existing scripts.

--numeric, -n

Show numerical addresses instead of trying to determine symbolic host, port or user names.

--numeric-hosts

shows numerical host addresses but does not affect the resolution of port or user names.

--numeric-ports

shows numerical port numbers but does not affect the resolution of host or user names.

--numeric-users

shows numerical user IDs but does not affect the resolution of host or port names.

--protocol=family, -A

Specifies the address families (perhaps better described as low level protocols) for which connections are to be shown. *family* is a comma (',') separated list of address family keywords like **inet**, **inet6**, **unix**, **ipx**, **ax25**, **netrom**, **econet**, **ddp**, and **bluetooth**. This has the same effect as using the **--inet|-4**, **--inet6|-6**, **--unix|-x**, **--ipx**, **--ax25**, **--netrom**, **--ddp**, and **--bluetooth** options.

The address family **inet** (Iv4) includes raw, udp, udplite and tcp protocol sockets.

The address family **bluetooth** (Iv4) includes l2cap and rfcomm protocol sockets.

-c, --continuous

This will cause **netstat** to print the selected information every second continuously.

-e, --extend

Display additional information. Use this option twice for maximum detail.

-o, --timers

Include information related to networking timers.

-p, --program

Show the PID and name of the program to which each socket belongs.

-l, --listening

Show only listening sockets. (These are omitted by default.)

-a, --all

Show both listening and non-listening sockets. With the **--interfaces** option, show interfaces that are not up

-F

Print routing information from the FIB. (This is the default.)

-C

Print routing information from the route cache.

OUTPUT

Active Internet connections (TCP, UDP, UDPLite, raw)**Proto**

The protocol (tcp, udp, udpl, raw) used by the socket.

Recv-Q

Established: The count of bytes not copied by the user program connected to this socket. Listening: Since Kernel 2.6.18 this column contains the current syn backlog.

Send-Q

Established: The count of bytes not acknowledged by the remote host. Listening: Since Kernel 2.6.18 this column contains the maximum size of the syn backlog.

Local Address

Address and port number of the local end of the socket. Unless the **--numeric (-n)** option is specified, the socket address is resolved to its canonical host name (FQDN), and the port number is translated into the corresponding service name.

Foreign Address

Address and port number of the remote end of the socket. Analogous to "Local Address".

State

The state of the socket. Since there are no states in raw mode and usually no states used in UDP and UDPLite, this column may be left blank. Normally this can be one of several values:

ESTABLISHED

The socket has an established connection.

SYN_SENT

The socket is actively attempting to establish a connection.

SYN_RECV

A connection request has been received from the network.

FIN_WAIT1

The socket is closed, and the connection is shutting down.

FIN_WAIT2

Connection is closed, and the socket is waiting for a shutdown from the remote end.

TIME_WAIT

The socket is waiting after close to handle packets still in the network.

CLOSE

The socket is not being used.

CLOSE_WAIT

The remote end has shut down, waiting for the socket to close.

LAST_ACK

The remote end has shut down, and the socket is closed. Waiting for acknowledgement.

LISTEN

The socket is listening for incoming connections. Such sockets are not included in the output unless you specify the **--listening (-l)** or **--all (-a)** option.

CLOSING

Both sockets are shut down but we still don't have all our data sent.

UNKNOWN

The state of the socket is unknown.

User

The username or the user id (UID) of the owner of the socket.

PID/Program name

Slash-separated pair of the process id (PID) and process name of the process that owns the socket. **--program** causes this column to be included. You will also need *superuser* privileges to see this information on sockets you don't own. This identification information is not yet available for IPX sockets.

Timer

(this needs to be written)

Active UNIX domain Sockets

Proto

The protocol (usually unix) used by the socket.

RefCnt

The reference count (i.e. attached processes via this socket).

Flags

The flags displayed is SO_ACCEPTON (displayed as **ACC**), SO_WAITDATA (**W**) or SO_NOSPACE (**N**). SO_ACCECPTON is used on unconnected sockets if their corresponding processes are waiting for a connect request. The other flags are not of normal interest.

Type

There are several types of socket access:

SOCK_DGRAM

The socket is used in Datagram (connectionless) mode.

SOCK_STREAM

This is a stream (connection) socket.

SOCK_RAW

The socket is used as a raw socket.

SOCK_RDM

This one serves reliably-delivered messages.

SOCK_SEQPACKET

This is a sequential packet socket.

SOCK_PACKET

Raw interface access socket.

UNKNOWN

Who ever knows what the future will bring us - just fill in here :-)

State

This field will contain one of the following Keywords:

FREE

The socket is not allocated

LISTENING

The socket is listening for a connection request. Such sockets are only included in the output if you specify the **--listening (-l)** or **--all (-a)** option.

CONNECTING

The socket is about to establish a connection.

CONNECTED

The socket is connected.

DISCONNECTING

The socket is disconnecting.

(empty) The socket is not connected to another one.

UNKNOWN

This state should never happen.

PID/Program name

Process ID (PID) and process name of the process that has the socket open. More info available in **Active Internet connections** section written above.

Path

This is the path name as which the corresponding processes attached to the socket.

Active IPX sockets

(this needs to be done by somebody who knows it)

Active NET/ROM sockets

(this needs to be done by somebody who knows it)

Active AX.25 sockets

(this needs to be done by somebody who knows it)

FILES

/etc/services -- The services translation file

/proc -- Mount point for the proc filesystem, which gives access to kernel status information via the following files.

/proc/net/dev -- device information

/proc/net/raw -- raw socket information

/proc/net/tcp -- TCP socket information

/proc/net/udp -- UDP socket information

/proc/net/udplite -- UDPLite socket information

/proc/net/igmp -- IGMP multicast information

/proc/net/unix -- Unix domain socket information

/proc/net/ipx -- IPX socket information

/proc/net/ax25 -- AX25 socket information

/proc/net/appletalk -- DDP (appletalk) socket information

/proc/net/nr -- NET/ROM socket information

/proc/net/route -- IP routing information

/proc/net/ax25_route -- AX25 routing information

/proc/net/px_route -- IPX routing information

/proc/net/nr_nodes -- NET/ROM nodelist

/proc/net/nr_neigh -- NET/ROM neighbours

/proc/net/ip_masquerade -- masqueraded connections

/sys/kernel/debug/bluetooth/l2cap -- Bluetooth L2CAP information

/sys/kernel/debug/bluetooth/rfcomm -- Bluetooth serial connections

/proc/net/snmp -- statistics

SEE ALSO

route(8), ifconfig(8), iptables(8), proc(5) ss(8) ip(8)

BUGS

Occasionally strange information may appear if a socket changes as it is viewed. This is unlikely to occur.

AUTHORS

The netstat user interface was written by Fred Baumgarten <dc6iq@insu1.etec.uni-karlsruhe.de>, the man page basically by Matt Welsh <mdw@tc.cornell.edu>. It was updated by Alan Cox <Alan.Cox@linux.org>, updated again by Tuan Hoang <tqhoang@bigfoot.com>. The man page and the command included in the net-tools package is totally rewritten by Bernd Eckenfels <ecki@linux.de>. UDPLite options were added by Brian Micek <bmicek@gmail.com>

nfsmount.conf(5) - Linux man page

Name

nfsmount.conf - Configuration file for NFS mounts

Synopsis

Configuration file for NFS mounts that allows options to be set globally, per server or per mount point.

Description

The configuration file is made up of multiple sections followed by variables associated with that section. A section is defined by a string enclosed by [and] branches. Variables are assignment statements that assign values to particular variables using the = operator, as in **Proto=Tcp**. Sections are broken up into three basic categories: Global options, Server options and Mount Point options.

[NFSMount_Global_Options] - This statically named section defines all of the global mount options that can be applied to every NFS mount.

[Server "Server_Name"] - This section defines all the mount options that should be used on mounts to a particular NFS server. The "Server_Name" strings needs to be surrounded by "" and be an exact match of the server name used in the **mount** command.

[MountPoint "Mount_Point"] - This section defines all the mount options that should be used on a particular mount point. The "Mount_Point" string needs to be surrounded by "" and be an exact match of the mount point used in the **mount** command.

Examples

These are some example lines of how sections and variables are defined in the configuration file.

```
[ NFSMount_Global_Options ]
Proto=Tcp
```

The TCP protocol will be used on every NFS mount.

```
[ Server "nfsserver.foo.com" ]
rsize=32k
wsize=32k
```

A 33k (32768 bytes) block size will be used as the read and write size on all mounts to the 'nfsserver.foo.com' server.

```
[ MountPoint "/export/home" ]
Background=True
```

All mounts to the '/export/home' export will be performed in the background (i.e. done asynchronously).

Files

/etc/nfsmount.conf

Default NFS mount configuration file

See Also

[nfs\(5\)](#), [mount\(8\)](#),

nfsstat(8) - Linux man page

Name

nfsstat - list NFS statistics

Synopsis

nfsstat [*OPTION*]...

Description

The **nfsstat** displays statistics kept about NFS client and server activity.

Options

-s, --server

Print only server-side statistics. The default is to print both server and client statistics.

-c, --client

Print only client-side statistics.

-n, --nfs

Print only NFS statistics. The default is to print both NFS and RPC information.

-2

Print only NFS v2 statistics. The default is to only print information about the versions of **NFS** that have non-zero counts.

-3

Print only NFS v3 statistics. The default is to only print information about the versions of **NFS** that have non-zero counts.

-4

Print only NFS v4 statistics. The default is to only print information about the versions of **NFS** that have non-zero counts.

-m, --mounts

Print information about each of the mounted **NFS** file systems.

If this option is used, all other options are ignored.

-r, --rpc

Print only RPC statistics.

-o *facility*

Display statistics for the specified facility, which must be one of:

nfs

NFS protocol information, split up by RPC call.

rpc

General RPC information.

net

Network layer statistics, such as the number of received packets, number of TCP connections, etc.

fh

Usage information on the server's file handle cache, including the total number of lookups, and the number of hits and misses.

rc

Usage information on the server's request reply cache, including the total number of lookups, and the number of hits and misses.

all

Display all of the above facilities.

-v, --verbose

This is equivalent to **-o all**.

-l, --list

Print information in list form.

-S, --since file

Instead of printing current statistics, **nfsstat** imports statistics from *file* and displays the difference between those and the current statistics. Valid input *files* may be in the form of **/proc/net/rpc/nfs** (raw client stats), **/proc/net/rpc/nfsd** (raw server stats), or saved output from **nfsstat** itself (client and/or server stats). Any statistics missing from a saved **nfsstat** output *file* are treated as zeroes.

-Z[interval], --sleep=[interval]

Instead of printing current statistics and immediately exiting, **nfsstat** takes a snapshot of the current statistics and pauses until it receives **SIGINT** (typically from **Ctrl-C**), at which point it takes another snapshot and displays the difference between the two. If *interval* is specified, **nfsstat** will print the number of **NFS** calls made since the previous report. Stats will be printed repeatedly every *interval* seconds.

Examples

nfsstat -o all -234

Show all information about all versions of **NFS**.

nfsstat --verbose -234

Same as above.

nfsstat -o all

Show all information about active versions of **NFS**.

nfsstat --nfs --server -3

Show statistics for **NFS** version 3 server.

nfsstat -m

Show information about mounted **NFS** filesystems.

Display

The **Flags** output from the **-m** option is the same as the flags give to the **mount** command.

Files

/proc/net/rpc/nfsd

procfs-based interface to kernel NFS server statistics.

/proc/net/rpc/nfs

procfs-based interface to kernel NFS client statistics.

/proc/mounts

procfs-based interface to the mounted filesystems.

See Also

[rpc.nfsd\(8\)](#). [nfs\(5\)](#).

Bugs

The default output has been changed. To get the old default output you must run **nfsstat -auto -2**.

The function of the **-v** and **-a** options have changed. The **-a** option is now reserved for future use. The **-v** does what the **-a** option used to do, and the new **-[234]** options replace the **-v** option.

The **Display** section should be more complete.

Further bugs can be found or reported at <http://nfs.sf.net/>.

Author

Olaf Kirch, <okir@suse.de>

Referenced By

[mountstats\(8\)](#), [nfsd\(7\)](#), [nfsd\(8\)](#), [nfsiostat\(8\)](#)

NAME

nice – run a program with modified scheduling priority

SYNOPSIS

nice [*OPTION*] [*COMMAND* [*ARG*]...]

DESCRIPTION

Run *COMMAND* with an adjusted niceness, which affects process scheduling. With no *COMMAND*, print the current niceness. Niceness values range from **-20** (most favorable to the process) to 19 (least favorable to the process).

Mandatory arguments to long options are mandatory for short options too.

-n, --adjustment=N

add integer N to the niceness (default 10)

--help display this help and exit

--version

output version information and exit

NOTE: your shell may have its own version of nice, which usually supersedes the version described here. Please refer to your shell's documentation for details about the options it supports.

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

nice(2), **renice**(1)

Full documentation <<https://www.gnu.org/software/coreutils/nice>>
or available locally via: info '(coreutils) nice invocation'

NAME

nice – change process priority

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int nice(int inc);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
nice():
_XOPEN_SOURCE
|| /* Since glibc 2.19: */ _DEFAULT_SOURCE
|| /* glibc <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE
```

DESCRIPTION

nice() adds *inc* to the nice value for the calling thread. (A higher nice value means a lower priority.)

The range of the nice value is +19 (low priority) to -20 (high priority). Attempts to set a nice value outside the range are clamped to the range.

Traditionally, only a privileged process could lower the nice value (i.e., set a higher priority). However, since Linux 2.6.12, an unprivileged process can decrease the nice value of a target process that has a suitable **RLIMIT_NICE** soft limit; see **getrlimit(2)** for details.

RETURN VALUE

On success, the new nice value is returned (but see NOTES below). On error, -1 is returned, and *errno* is set to indicate the error.

A successful call can legitimately return -1. To detect an error, set *errno* to 0 before the call, and check whether it is nonzero after **nice()** returns -1.

ERRORS

EPERM

The calling process attempted to increase its priority by supplying a negative *inc* but has insufficient privileges. Under Linux, the**CAP_SYS_NICE** capability is required. (But see the discussion of the **RLIMIT_NICE** resource limit in **setrlimit(2)**.)

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD. However, the raw system call and (g)libc (earlier than glibc 2.2.4) return value is nonstandard, see below.

NOTES

For further details on the nice value, see **sched(7)**.

Note: the addition of the "autogroup" feature in Linux 2.6.38 means that the nice value no longer has its traditional effect in many circumstances. For details, see **sched(7)**.

C library/kernel differences

POSIX.1 specifies that **nice()** should return the new nice value. However, the raw Linux system call returns 0 on success. Likewise, the **nice()** wrapper function provided in glibc 2.2.3 and earlier returns 0 on success.

Since glibc 2.2.4, the **nice()** wrapper function provided by glibc provides conformance to POSIX.1 by calling **getpriority(2)** to obtain the new nice value, which is then returned to the caller.

SEE ALSO

nice(1), **renice(1)**, **fork(2)**, **getpriority(2)**, **getrlimit(2)**, **setpriority(2)**, **capabilities(7)**, **sched(7)**

NAME

nm-settings-nmcli – Description of settings and properties of NetworkManager connection profiles for nmcli

DESCRIPTION

NetworkManager is based on a concept of connection profiles, sometimes referred to as connections only. These connection profiles contain a network configuration. When NetworkManager activates a connection profile on a network device the configuration will be applied and an active network connection will be established. Users are free to create as many connection profiles as they see fit. Thus they are flexible in having various network configurations for different networking needs.

NetworkManager provides an API for configuring connection profiles, for activating them to configure the network, and inspecting the current network configuration. The command line tool *nmcli* is a client application to NetworkManager that uses this API. See **nmcli(1)** for details.

With commands like *nmcli connection add*, *nmcli connection modify* and *nmcli connection show*, connection profiles can be created, modified and inspected. A profile consists of properties. On D-Bus this follows the format as described by **nm-settings-dbus(5)**, while this manual page describes the settings format how they are expected by *nmcli*.

The settings and properties shown in tables below list all available connection configuration options. However, note that not all settings are applicable to all connection types. *nmcli* connection editor has also a built-in *describe* command that can display description of particular settings and properties of this page.

The *setting* and *property* can be abbreviated provided they are unique. The list below also shows aliases that can be used unqualified instead of the full name. For example *connection.interface-name* and *ifname* refer to the same property.

connection setting

General Connection Profile Settings.

Properties:

auth-retries

The number of retries for the authentication. Zero means to try indefinitely; -1 means to use a global default. If the global default is not set, the authentication retries for 3 times before failing the connection. Currently, this only applies to 802-1x authentication.

Format: int32

autoconnect

Alias: autoconnect

Whether or not the connection should be automatically connected by NetworkManager when the resources for the connection are available. TRUE to automatically activate the connection, FALSE to require manual intervention to activate the connection. Autoconnect happens when the circumstances are suitable. That means for example that the device is currently managed and not active. Autoconnect thus never replaces or competes with an already active profile. Note that autoconnect is not implemented for VPN profiles. See "secondaries" as an alternative to automatically connect VPN profiles.

Format: boolean

autoconnect-priority

The autoconnect priority in range -999 to 999. If the connection is set to autoconnect, connections with higher priority will be preferred. The higher number means higher priority. Defaults to 0. Note that this property only matters if there are more than one candidate profile to select for autoconnect. In case of equal priority, the profile used most recently is chosen.

Format: int32

autoconnect-retries

The number of times a connection should be tried when autoactivating before giving up. Zero means forever, -1 means the global default (4 times if not overridden). Setting this to 1 means to try activation only once before blocking autoconnect. Note that after a timeout, NetworkManager will try to autoconnect again.

Format: int32

autoconnect-slaves

Whether or not slaves of this connection should be automatically brought up when NetworkManager activates this connection. This only has a real effect for master connections. The properties "autoconnect", "autoconnect-priority" and "autoconnect-retries" are unrelated to this setting. The permitted values are: 0: leave slave connections untouched, 1: activate all the slave connections with this connection, -1: default. If -1 (default) is set, global connection.autoconnect-slaves is read to determine the real value. If it is default as well, this falls back to 0.

Format: NMSettingConnectionAutoconnectSlaves (int32)

dns-over-tls

Whether DNSOverTls (dns-over-tls) is enabled for the connection. DNSOverTls is a technology which uses TLS to encrypt dns traffic. The permitted values are: "yes" (2) use DNSOverTls and disabled fallback, "opportunistic" (1) use DNSOverTls but allow fallback to unencrypted resolution, "no" (0) don't ever use DNSOverTls. If unspecified "default" depends on the plugin used. Systemd-resolved uses global setting. This feature requires a plugin which supports DNSOverTls. Otherwise, the setting has no effect. One such plugin is dns-systemd-resolved.

Format: int32

gateway-ping-timeout

If greater than zero, delay success of IP addressing until either the timeout is reached, or an IP gateway replies to a ping.

Format: uint32

id

Alias: con-name

A human readable unique identifier for the connection, like "Work Wi-Fi" or "T-Mobile 3G".

Format: string

interface-name

Alias: ifname

The name of the network interface this connection is bound to. If not set, then the connection can be attached to any interface of the appropriate type (subject to restrictions imposed by other settings). For software devices this specifies the name of the created device. For connection types where interface names cannot easily be made persistent (e.g. mobile broadband or USB Ethernet), this property should not be used. Setting this property restricts the interfaces a connection can be used with, and if interface names change or are reordered the connection may be applied to the wrong interface.

Format: string

lldp

Whether LLDP is enabled for the connection.

Format: int32

llmnr

Whether Link–Local Multicast Name Resolution (LLMNR) is enabled for the connection. LLMNR is a protocol based on the Domain Name System (DNS) packet format that allows both IPv4 and IPv6 hosts to perform name resolution for hosts on the same local link. The permitted values are: "yes" (2) register hostname and resolving for the connection, "no" (0) disable LLMNR for the interface, "resolve" (1) do not register hostname but allow resolving of LLMNR host names If unspecified, "default" ultimately depends on the DNS plugin (which for systemd–resolved currently means "yes"). This feature requires a plugin which supports LLMNR. Otherwise, the setting has no effect. One such plugin is dns–systemd–resolved.

Format: int32

master

Alias: master

Interface name of the master device or UUID of the master connection.

Format: string

mdns

Whether mDNS is enabled for the connection. The permitted values are: "yes" (2) register hostname and resolving for the connection, "no" (0) disable mDNS for the interface, "resolve" (1) do not register hostname but allow resolving of mDNS host names and "default" (-1) to allow lookup of a global default in NetworkManager.conf. If unspecified, "default" ultimately depends on the DNS plugin (which for systemd–resolved currently means "no"). This feature requires a plugin which supports mDNS. Otherwise, the setting has no effect. One such plugin is dns–systemd–resolved.

Format: int32

metered

Whether the connection is metered. When updating this property on a currently activated connection, the change takes effect immediately.

Format: NMMetered (int32)

mud–url

If configured, set to a Manufacturer Usage Description (MUD) URL that points to manufacturer–recommended network policies for IoT devices. It is transmitted as a DHCPv4 or DHCPv6 option. The value must be a valid URL starting with "https://". The special value "none" is allowed to indicate that no MUD URL is used. If the per–profile value is unspecified (the default), a global connection default gets consulted. If still unspecified, the ultimate default is "none".

Format: string

multi–connect

Specifies whether the profile can be active multiple times at a particular moment. The value is of type NMConnectionMultiConnect.

Format: int32

permissions

An array of strings defining what access a given user has to this connection. If this is NULL or empty, all users are allowed to access this connection; otherwise users are allowed if and only if they are in this list. When this is not empty, the connection can be active only when one of the specified users is logged into an active session. Each entry is of the form "[type]:[id]:[reserved]"; for example, "user:dcbw:blah". At this time only the "user" [type] is allowed. Any other values are ignored and reserved for future use. [id] is the username that this permission refers to, which may not contain the ":" character. Any [reserved] information present must be ignored and is reserved for future use. All of [type], [id], and [reserved] must be valid UTF–8.

Format: array of string

read-only

FALSE if the connection can be modified using the provided settings service's D-Bus interface with the right privileges, or TRUE if the connection is read-only and cannot be modified.

Format: boolean

secondaries

List of connection UUIDs that should be activated when the base connection itself is activated. Currently, only VPN connections are supported.

Format: array of string

slave-type

Alias: slave-type

Setting name of the device type of this slave's master connection (eg, "bond"), or NULL if this connection is not a slave.

Format: string

stable-id

This represents the identity of the connection used for various purposes. It allows to configure multiple profiles to share the identity. Also, the stable-id can contain placeholders that are substituted dynamically and deterministically depending on the context. The stable-id is used for generating IPv6 stable private addresses with ipv6.addr-gen-mode=stable-privacy. It is also used to seed the generated cloned MAC address for ethernet.cloned-mac-address=stable and wifi.cloned-mac-address=stable. It is also used as DHCP client identifier with ipv4.dhcp-client-id=stable and to derive the DHCP DUID with ipv6.dhcp-duid=stable-[llt, ll, uuid]. Note that depending on the context where it is used, other parameters are also seeded into the generation algorithm. For example, a per-host key is commonly also included, so that different systems end up generating different IDs. Or with ipv6.addr-gen-mode=stable-privacy, also the device's name is included, so that different interfaces yield different addresses. The per-host key is the identity of your machine and stored in /var/lib/NetworkManager/secret-key. The '\$' character is treated special to perform dynamic substitutions at runtime. Currently, supported are "\${CONNECTION}", "\${DEVICE}", "\${MAC}", "\${BOOT}", "\${RANDOM}". These effectively create unique IDs per-connection, per-device, per-boot, or every time. Note that "\${DEVICE}" corresponds to the interface name of the device and "\${MAC}" is the permanent MAC address of the device. Any unrecognized patterns following '\$' are treated verbatim, however are reserved for future use. You are thus advised to avoid '\$' or escape it as "\$\$". For example, set it to "\${CONNECTION}-\${BOOT}-\${DEVICE}" to create a unique id for this connection that changes with every reboot and differs depending on the interface where the profile activates. If the value is unset, a global connection default is consulted. If the value is still unset, the default is similar to "\${CONNECTION}" and uses a unique, fixed ID for the connection.

Format: string

timestamp

The time, in seconds since the Unix Epoch, that the connection was last _successfully_ fully activated. NetworkManager updates the connection timestamp periodically when the connection is active to ensure that an active connection has the latest timestamp. The property is only meant for reading (changes to this property will not be preserved).

Format: uint64

type

Alias: type

Base type of the connection. For hardware-dependent connections, should contain the setting name of the hardware-type specific setting (ie, "802-3-ethernet" or "802-11-wireless" or "bluetooth", etc), and for non-hardware dependent connections like VPN or otherwise, should contain the setting name of that setting type (ie, "vpn" or "bridge", etc).

Format: string

uuid

A universally unique identifier for the connection, for example generated with libuuid. It should be assigned when the connection is created, and never changed as long as the connection still applies to the same network. For example, it should not be changed when the "id" property or NMSettingIP4Config changes, but might need to be re-created when the Wi-Fi SSID, mobile broadband network provider, or "type" property changes. The UUID must be in the format "2815492f-7e56-435e-b2e9-246bd7cdc664" (ie, contains only hexadecimal characters and "-").

Format: string

wait-device-timeout

Timeout in milliseconds to wait for device at startup. During boot, devices may take a while to be detected by the driver. This property will cause NetworkManager-wait-online.service and nm-online to give the device a chance to appear. This works by waiting for the given timeout until a compatible device for the profile is available and managed. The value 0 means no wait time. The default value is -1, which currently has the same meaning as no wait time.

Format: int32

zone

The trust level of a the connection. Free form case-insensitive string (for example "Home", "Work", "Public"). NULL or unspecified zone means the connection will be placed in the default zone as defined by the firewall. When updating this property on a currently activated connection, the change takes effect immediately.

Format: string

6lowpan setting

6LoWPAN Settings.

Properties:

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this 6LowPAN interface should be created.

Format: string

802-1x setting

IEEE 802.1x Authentication Settings.

Properties:

altsubject-matches

List of strings to be matched against the altSubjectName of the certificate presented by the authentication server. If the list is empty, no verification of the server certificate's altSubjectName is performed.

Format: array of string

anonymous-identity

Anonymous identity string for EAP authentication methods. Used as the unencrypted identity with EAP types that support different tunneled identity like EAP-TTLS.

Format: string

auth-timeout

A timeout for the authentication. Zero means the global default; if the global default is not set, the authentication timeout is 25 seconds.

Format: int32

ca-cert

Contains the CA certificate if used by the EAP method specified in the "eap" property. Certificate data is specified using a "scheme"; three are currently supported: blob, path and pkcs#11 URL. When using the blob scheme this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended. Note that enabling NMSetting8021x:system-ca-certs will override this setting to use the built-in path, if the built-in path is not a directory.

Format: byte array

ca-cert-password

The password used to access the CA certificate stored in "ca-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

ca-cert-password-flags

Flags indicating how to handle the "ca-cert-password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

ca-path

UTF-8 encoded path to a directory containing PEM or DER formatted certificates to be added to the verification chain in addition to the certificate specified in the "ca-cert" property. If NMSetting8021x:system-ca-certs is enabled and the built-in CA path is an existing directory, then this setting is ignored.

Format: string

client-cert

Contains the client certificate if used by the EAP method specified in the "eap" property. Certificate data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme (which is backwards compatible with NM 0.7.x) this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte.

Format: byte array

client-cert-password

The password used to access the client certificate stored in "client-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

client-cert-password-flags

Flags indicating how to handle the "client-cert-password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

domain-match

Constraint for server domain name. If set, this list of FQDNs is used as a match requirement for dNSName element(s) of the certificate presented by the authentication server. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using the same comparison. Multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

domain-suffix-match

Constraint for server domain name. If set, this FQDN is used as a suffix match requirement for dNSName element(s) of the certificate presented by the authentication server. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using same suffix match comparison. Since version 1.24, multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

eap

The allowed EAP method to be used when authenticating to the network with 802.1x. Valid methods are: "leap", "md5", "tls", "peap", "ttls", "pwd", and "fast". Each method requires different configuration using the properties of this setting; refer to wpa_supplicant documentation for the allowed combinations.

Format: array of string

identity

Identity string for EAP authentication methods. Often the user's user or login name.

Format: string

optional

Whether the 802.1X authentication is optional. If TRUE, the activation will continue even after a timeout or an authentication failure. Setting the property to TRUE is currently allowed only for Ethernet connections. If set to FALSE, the activation can continue only after a successful authentication.

Format: boolean

pac-file

UTF-8 encoded file path containing PAC for EAP-FAST.

Format: string

password

UTF-8 encoded password used for EAP authentication methods. If both the "password" property and the "password-raw" property are specified, "password" is preferred.

Format: string

password-flags

Flags indicating how to handle the "password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

password–raw

Password used for EAP authentication methods, given as a byte array to allow passwords in other encodings than UTF-8 to be used. If both the "password" property and the "password–raw" property are specified, "password" is preferred.

Format: byte array

password–raw–flags

Flags indicating how to handle the "password–raw" property. See the section called “Secret flag types:” for flag values.

Format: NMSettingSecretFlags (uint32)

phase1–auth–flags

Specifies authentication flags to use in "phase 1" outer authentication using NMSetting8021xAuthFlags options. The individual TLS versions can be explicitly disabled. If a certain TLS disable flag is not set, it is up to the supplicant to allow or forbid it. The TLS options map to tls_disable_tlv1_x settings. See the wpa_supplicant documentation for more details.

Format: uint32

phase1–fast–provisioning

Enables or disables in-line provisioning of EAP–FAST credentials when FAST is specified as the EAP method in the "eap" property. Recognized values are "0" (disabled), "1" (allow unauthenticated provisioning), "2" (allow authenticated provisioning), and "3" (allow both authenticated and unauthenticated provisioning). See the wpa_supplicant documentation for more details.

Format: string

phase1–peaplabel

Forces use of the new PEAP label during key derivation. Some RADIUS servers may require forcing the new PEAP label to interoperate with PEAPv1. Set to "1" to force use of the new PEAP label. See the wpa_supplicant documentation for more details.

Format: string

phase1–peapver

Forces which PEAP version is used when PEAP is set as the EAP method in the "eap" property. When unset, the version reported by the server will be used. Sometimes when using older RADIUS servers, it is necessary to force the client to use a particular PEAP version. To do so, this property may be set to "0" or "1" to force that specific PEAP version.

Format: string

phase2–altsubject–matches

List of strings to be matched against the altSubjectName of the certificate presented by the authentication server during the inner "phase 2" authentication. If the list is empty, no verification of the server certificate's altSubjectName is performed.

Format: array of string

phase2–auth

Specifies the allowed "phase 2" inner authentication method when an EAP method that uses an inner TLS tunnel is specified in the "eap" property. For TTLS this property selects one of the supported non–EAP inner methods: "pap", "chap", "mschap", "mschapv2" while "phase2–autheap" selects an EAP inner method. For PEAP this selects an inner EAP method, one of: "gtc", "otp", "md5" and "tls". Each "phase 2" inner method requires specific parameters for successful authentication; see the

wpa_supplicant documentation for more details. Both "phase2-auth" and "phase2-autheap" cannot be specified.

Format: string

phase2-autheap

Specifies the allowed "phase 2" inner EAP-based authentication method when TLS is specified in the "eap" property. Recognized EAP-based "phase 2" methods are "md5", "mschapv2", "otp", "gtc", and "tls". Each "phase 2" inner method requires specific parameters for successful authentication; see the wpa_supplicant documentation for more details.

Format: string

phase2-ca-cert

Contains the "phase 2" CA certificate if used by the EAP method specified in the "phase2-auth" or "phase2-autheap" properties. Certificate data is specified using a "scheme"; three are currently supported: blob, path and pkcs#11 URL. When using the blob scheme this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended. Note that enabling NMSetting8021x:system-ca-certs will override this setting to use the built-in path, if the built-in path is not a directory.

Format: byte array

phase2-ca-cert-password

The password used to access the "phase2" CA certificate stored in "phase2-ca-cert" property. Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

phase2-ca-cert-password-flags

Flags indicating how to handle the "phase2-ca-cert-password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

phase2-ca-path

UTF-8 encoded path to a directory containing PEM or DER formatted certificates to be added to the verification chain in addition to the certificate specified in the "phase2-ca-cert" property. If NMSetting8021x:system-ca-certs is enabled and the built-in CA path is an existing directory, then this setting is ignored.

Format: string

phase2-client-cert

Contains the "phase 2" client certificate if used by the EAP method specified in the "phase2-auth" or "phase2-autheap" properties. Certificate data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme (which is backwards compatible with NM 0.7.x) this property should be set to the certificate's DER encoded data. When using the path scheme, this property should be set to the full UTF-8 encoded path of the certificate, prefixed with the string "file://" and ending with a terminating NUL byte. This property can be unset even if the EAP method supports CA certificates, but this allows man-in-the-middle attacks and is NOT recommended.

Format: byte array

phase2-client-cert-password

The password used to access the "phase2" client certificate stored in "phase2-client-cert" property.

Only makes sense if the certificate is stored on a PKCS#11 token that requires a login.

Format: string

phase2-client-cert-password-flags

Flags indicating how to handle the "phase2-client-cert-password" property. See the section called "Secret flag types." for flag values.

Format: NMSettingSecretFlags (uint32)

phase2-domain-match

Constraint for server domain name. If set, this list of FQDNs is used as a match requirement for dNSName element(s) of the certificate presented by the authentication server during the inner "phase 2" authentication. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using the same comparison. Multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

phase2-domain-suffix-match

Constraint for server domain name. If set, this FQDN is used as a suffix match requirement for dNSName element(s) of the certificate presented by the authentication server during the inner "phase 2" authentication. If a matching dNSName is found, this constraint is met. If no dNSName values are present, this constraint is matched against SubjectName CN using same suffix match comparison. Since version 1.24, multiple valid FQDNs can be passed as a ";" delimited list.

Format: string

phase2-private-key

Contains the "phase 2" inner private key when the "phase2-auth" or "phase2-autheap" property is set to "tls". Key data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme and private keys, this property should be set to the key's encrypted PEM encoded data. When using private keys with the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte. When using PKCS#12 format private keys and the blob scheme, this property should be set to the PKCS#12 data and the "phase2-private-key-password" property must be set to password used to decrypt the PKCS#12 certificate and key. When using PKCS#12 files and the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte, and as with the blob scheme the "phase2-private-key-password" property must be set to the password used to decode the PKCS#12 private key and certificate.

Format: byte array

phase2-private-key-password

The password used to decrypt the "phase 2" private key specified in the "phase2-private-key" property when the private key either uses the path scheme, or is a PKCS#12 format key.

Format: string

phase2-private-key-password-flags

Flags indicating how to handle the "phase2-private-key-password" property. See the section called "Secret flag types." for flag values.

Format: NMSettingSecretFlags (uint32)

phase2-subject-match

Substring to be matched against the subject of the certificate presented by the authentication server during the inner "phase 2" authentication. When unset, no verification of the authentication server

certificate's subject is performed. This property provides little security, if any, and its use is deprecated in favor of NMSetting8021x:phase2-domain-suffix-match.

Format: string

pin

PIN used for EAP authentication methods.

Format: string

pin-flags

Flags indicating how to handle the "pin" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

private-key

Contains the private key when the "eap" property is set to "tls". Key data is specified using a "scheme"; two are currently supported: blob and path. When using the blob scheme and private keys, this property should be set to the key's encrypted PEM encoded data. When using private keys with the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte. When using PKCS#12 format private keys and the blob scheme, this property should be set to the PKCS#12 data and the "private-key-password" property must be set to password used to decrypt the PKCS#12 certificate and key. When using PKCS#12 files and the path scheme, this property should be set to the full UTF-8 encoded path of the key, prefixed with the string "file://" and ending with a terminating NUL byte, and as with the blob scheme the "private-key-password" property must be set to the password used to decode the PKCS#12 private key and certificate. WARNING: "private-key" is not a "secret" property, and thus unencrypted private key data using the BLOB scheme may be readable by unprivileged users. Private keys should always be encrypted with a private key password to prevent unauthorized access to unencrypted private key data.

Format: byte array

private-key-password

The password used to decrypt the private key specified in the "private-key" property when the private key either uses the path scheme, or if the private key is a PKCS#12 format key.

Format: string

private-key-password-flags

Flags indicating how to handle the "private-key-password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

subject-match

Substring to be matched against the subject of the certificate presented by the authentication server. When unset, no verification of the authentication server certificate's subject is performed. This property provides little security, if any, and its use is deprecated in favor of NMSetting8021x:domain-suffix-match.

Format: string

system-ca-certs

When TRUE, overrides the "ca-path" and "phase2-ca-path" properties using the system CA directory specified at configure time with the --system-ca-path switch. The certificates in this directory are added to the verification chain in addition to any certificates specified by the "ca-cert" and "phase2-ca-cert" properties. If the path provided with --system-ca-path is rather a file name (bundle

of trusted CA certificates), it overrides "ca-cert" and "phase2-ca-cert" properties instead (sets ca_cert/ca_cert2 options for wpa_supplicant).

Format: boolean

adsl setting

ADSL Settings.

Properties:

encapsulation

Alias: encapsulation

Encapsulation of ADSL connection. Can be "vcmux" or "llc".

Format: string

password

Alias: password

Password used to authenticate with the ADSL service.

Format: string

password-flags

Flags indicating how to handle the "password" property. See the section called “Secret flag types:” for flag values.

Format: NMSettingSecretFlags (uint32)

protocol

Alias: protocol

ADSL connection protocol. Can be "pppoa", "pppoe" or "ipoatm".

Format: string

username

Alias: username

Username used to authenticate with the ADSL service.

Format: string

vci

VCI of ADSL connection

Format: uint32

vpi

VPI of ADSL connection

Format: uint32

bluetooth setting

Bluetooth Settings.

Properties:

bdaddr

Alias: addr

The Bluetooth address of the device.

Format: byte array

type

Alias: bt-type

Either "dun" for Dial-Up Networking connections or "panu" for Personal Area Networking connections to devices supporting the NAP profile.

Format: string

bond setting

Bonding Settings.

Properties:

options

Dictionary of key/value pairs of bonding options. Both keys and values must be strings. Option names must contain only alphanumeric characters (ie, [a-zA-Z0-9]).

Format: dict of string to string

bridge setting

Bridging Settings.

Properties:

ageing-time

Alias: ageing-time

The Ethernet MAC address aging time, in seconds.

Format: uint32

forward-delay

Alias: forward-delay

The Spanning Tree Protocol (STP) forwarding delay, in seconds.

Format: uint32

group-address

If specified, The MAC address of the multicast group this bridge uses for STP. The address must be a link-local address in standard Ethernet MAC address format, ie an address of the form 01:80:C2:00:00:0X, with X in [0, 4..F]. If not specified the default value is 01:80:C2:00:00:00.

Format: byte array

group-forward-mask

Alias: group-forward-mask

A mask of group addresses to forward. Usually, group addresses in the range from 01:80:C2:00:00:00 to 01:80:C2:00:00:0F are not forwarded according to standards. This property is a mask of 16 bits, each corresponding to a group address in that range that must be forwarded. The mask can't have bits 0, 1 or 2 set because they are used for STP, MAC pause frames and LACP.

Format: uint32

hello-time

Alias: hello-time

The Spanning Tree Protocol (STP) hello time, in seconds.

Format: uint32

mac-address

Alias: mac

If specified, the MAC address of bridge. When creating a new bridge, this MAC address will be set. If this field is left unspecified, the "ethernet.cloned-mac-address" is referred instead to generate the initial MAC address. Note that setting "ethernet.cloned-mac-address" anyway overwrites the MAC address of the bridge later while activating the bridge. Hence, this property is deprecated. Deprecated: 1

Format: byte array

max-age

Alias: max-age

The Spanning Tree Protocol (STP) maximum message age, in seconds.

Format: uint32

multicast-hash-max

Set maximum size of multicast hash table (value must be a power of 2).

Format: uint32

multicast-last-member-count

Set the number of queries the bridge will send before stopping forwarding a multicast group after a "leave" message has been received.

Format: uint32

multicast-last-member-interval

Set interval (in deciseconds) between queries to find remaining members of a group, after a "leave" message is received.

Format: uint64

multicast-membership-interval

Set delay (in deciseconds) after which the bridge will leave a group, if no membership reports for this group are received.

Format: uint64

multicast-querier

Enable or disable sending of multicast queries by the bridge. If not specified the option is disabled.

Format: boolean

multicast-querier-interval

If no queries are seen after this delay (in deciseconds) has passed, the bridge will start to send its own queries.

Format: uint64

multicast-query-interval

Interval (in deciseconds) between queries sent by the bridge after the end of the startup phase.

Format: uint64

multicast-query-response-interval

Set the Max Response Time/Max Response Delay (in deciseconds) for IGMP/MLD queries sent by the bridge.

Format: uint64

multicast-query-use-ifaddr

If enabled the bridge's own IP address is used as the source address for IGMP queries otherwise the default of 0.0.0.0 is used.

Format: boolean

multicast-router

Sets bridge's multicast router. Multicast-snooping must be enabled for this option to work. Supported values are: 'auto', 'disabled', 'enabled' to which kernel assigns the numbers 1, 0, and 2, respectively. If not specified the default value is 'auto' (1).

Format: string

multicast-snooping

Alias: multicast-snooping

Controls whether IGMP snooping is enabled for this bridge. Note that if snooping was automatically disabled due to hash collisions, the system may refuse to enable the feature until the collisions are resolved.

Format: boolean

multicast-startup-query-count

Set the number of IGMP queries to send during startup phase.

Format: uint32

multicast-startup-query-interval

Sets the time (in deciseconds) between queries sent out at startup to determine membership information.

Format: uint64

priority

Alias: priority

Sets the Spanning Tree Protocol (STP) priority for this bridge. Lower values are "better"; the lowest priority bridge will be elected the root bridge.

Format: uint32

stp

Alias: stp

Controls whether Spanning Tree Protocol (STP) is enabled for this bridge.

Format: boolean

vlan-default-pvid

The default PVID for the ports of the bridge, that is the VLAN id assigned to incoming untagged frames.

Format: uint32

vlan-filtering

Control whether VLAN filtering is enabled on the bridge.

Format: boolean

vlan-protocol

If specified, the protocol used for VLAN filtering. Supported values are: '802.1Q', '802.1ad'. If not specified the default value is '802.1Q'.

Format: string

vlan-stats-enabled

Controls whether per-VLAN stats accounting is enabled.

Format: boolean

vlans

Array of bridge VLAN objects. In addition to the VLANs specified here, the bridge will also have the default-pvid VLAN configured by the bridge.vlan-default-pvid property. In nmcli the VLAN list can be specified with the following syntax: \$vid [pvid] [untagged] [, \$vid [pvid] [untagged]]... where \$vid is either a single id between 1 and 4094 or a range, represented as a couple of ids separated by a dash.

Format: array of vardict

bridge-port setting

Bridge Port Settings.

Properties:

hairpin-mode

Alias: hairpin

Enables or disables "hairpin mode" for the port, which allows frames to be sent back out through the port the frame was received on.

Format: boolean

path-cost

Alias: path-cost

The Spanning Tree Protocol (STP) port cost for destinations via this port.

Format: uint32

priority

Alias: priority

The Spanning Tree Protocol (STP) priority of this bridge port.

Format: uint32

vlans

Array of bridge VLAN objects. In addition to the VLANs specified here, the port will also have the default-pvid VLAN configured on the bridge by the bridge.vlan-default-pvid property. In nmcli the VLAN list can be specified with the following syntax: \$vid [pvid] [untagged] [, \$vid [pvid] [untagged]]... where \$vid is either a single id between 1 and 4094 or a range, represented as a couple of ids separated by a dash.

Format: array of vardict

cdma setting

CDMA-based Mobile Broadband Settings.

Properties:

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple frames.

Format: uint32

number

The number to dial to establish the connection to the CDMA-based mobile broadband network, if any. If not specified, the default number (#777) is used when required.

Format: string

password

Alias: password

The password used to authenticate with the network, if required. Many providers do not require a password, or accept any password. But if a password is required, it is specified here.

Format: string

password-flags

Flags indicating how to handle the "password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

username

Alias: user

The username used to authenticate with the network, if required. Many providers do not require a username, or accept any username. But if a username is required, it is specified here.

Format: string

dcb setting

Data Center Bridging Settings.

Properties:

app-fcoe-flags

Specifies the NMSettingDcbFlags for the DCB FCoE application. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-fcoe-mode

The FCoE controller mode; either "fabric" or "vn2vn". Since 1.34, NULL is the default and means "fabric". Before 1.34, NULL was rejected as invalid and the default was "fabric".

Format: string

app-fcoe-priority

The highest User Priority (0 – 7) which FCoE frames should use, or -1 for default priority. Only used when the "app-fcoe-flags" property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

app-fip-flags

Specifies the NMSettingDcbFlags for the DCB FIP application. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-fip-priority

The highest User Priority (0 – 7) which FIP frames should use, or -1 for default priority. Only used when the "app-fip-flags" property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

app-iscsi-flags

Specifies the NMSettingDcbFlags for the DCB iSCSI application. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

app-iscsi-priority

The highest User Priority (0 – 7) which iSCSI frames should use, or -1 for default priority. Only used when the "app-iscsi-flags" property includes the NM_SETTING_DCB_FLAG_ENABLE (0x1) flag.

Format: int32

priority-bandwidth

An array of 8 uint values, where the array index corresponds to the User Priority (0 – 7) and the value indicates the percentage of bandwidth of the priority's assigned group that the priority may use. The sum of all percentages for priorities which belong to the same group must total 100 percents.

Format: array of uint32

priority-flow-control

An array of 8 boolean values, where the array index corresponds to the User Priority (0 – 7) and the value indicates whether or not the corresponding priority should transmit priority pause.

Format: array of uint32

priority-flow-control-flags

Specifies the NMSettingDcbFlags for DCB Priority Flow Control (PFC). Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

priority-group-bandwidth

An array of 8 uint values, where the array index corresponds to the Priority Group ID (0 – 7) and the value indicates the percentage of link bandwidth allocated to that group. Allowed values are 0 – 100, and the sum of all values must total 100 percents.

Format: array of uint32

priority-group-flags

Specifies the NMSettingDcbFlags for DCB Priority Groups. Flags may be any combination of NM_SETTING_DCB_FLAG_ENABLE (0x1), NM_SETTING_DCB_FLAG_ADVERTISE (0x2), and NM_SETTING_DCB_FLAG_WILLING (0x4).

Format: NMSettingDcbFlags (uint32)

priority-group-id

An array of 8 uint values, where the array index corresponds to the User Priority (0 – 7) and the value indicates the Priority Group ID. Allowed Priority Group ID values are 0 – 7 or 15 for the unrestricted group.

Format: array of uint32

priority-strict-bandwidth

An array of 8 boolean values, where the array index corresponds to the User Priority (0 – 7) and the value indicates whether or not the priority may use all of the bandwidth allocated to its assigned group.

Format: array of uint32

priority-traffic-class

An array of 8 uint values, where the array index corresponds to the User Priority (0 – 7) and the value indicates the traffic class (0 – 7) to which the priority is mapped.

Format: array of uint32

ethtool setting

Ethtool Ethernet Settings.

Properties:

- coalesce-adaptive-rx**
- coalesce-adaptive-tx**
- coalesce-pkt-rate-high**
- coalesce-pkt-rate-low**
- coalesce-rx-frames**
- coalesce-rx-frames-high**
- coalesce-rx-frames-irq**
- coalesce-rx-frames-low**
- coalesce-rx-usecs**
- coalesce-rx-usecs-high**
- coalesce-rx-usecs-irq**
- coalesce-rx-usecs-low**
- coalesce-sample-interval**
- coalesce-stats-block-usecs**
- coalesce-tx-frames**
- coalesce-tx-frames-high**
- coalesce-tx-frames-irq**
- coalesce-tx-frames-low**
- coalesce-tx-usecs**
- coalesce-tx-usecs-high**
- coalesce-tx-usecs-irq**
- coalesce-tx-usecs-low**
- feature-esp-hw-offload**

feature-esp-tx-csum-hw-offload
feature-fcoe-mtu
feature-gro
feature-gso
feature-highdma
feature-hw-tc-offload
feature-l2-fwd-offload
feature-loopback
feature-lro
feature-macsec-hw-offload
feature-ntuple
feature-rx
feature-rx-all
feature-rx-fcs
feature-rx-gro-hw
feature-rx-gro-list
feature-rx-udp-gro-forwarding
feature-rx-udp_tunnel-port-offload
feature-rx-vlan-filter
feature-rx-vlan-stag-filter
feature-rx-vlan-stag-hw-parse
feature-rxhash
feature-rxvlan
feature-sg
feature-tls-hw-record
feature-tls-hw-rx-offload
feature-tls-hw-tx-offload
feature-tso
feature-tx
feature-tx-checksum-fcoe-crc
feature-tx-checksum-ip-generic
feature-tx-checksum-ipv4
feature-tx-checksum-ipv6
feature-tx-checksum-sctp
feature-tx-esp-segmentation
feature-tx-fcoe-segmentation
feature-tx-gre-csum-segmentation
feature-tx-gre-segmentation
feature-tx-gso-list

feature-tx-gso-partial
feature-tx-gso-robust
feature-tx-ipxip4-segmentation
feature-tx-ipxip6-segmentation
feature-tx-nocache-copy
feature-tx-scatter-gather
feature-tx-scatter-gather-fraglist
feature-tx-sctp-segmentation
feature-tx-tcp-ecn-segmentation
feature-tx-tcp-mangleid-segmentation
feature-tx-tcp-segmentation
feature-tx-tcp6-segmentation
feature-tx-tunnel-remcsum-segmentation
feature-tx-udp-segmentation
feature-tx-udp_tnl-csum-segmentation
feature-tx-udp_tnl-segmentation
feature-tx-vlan-stag-hw-insert
feature-txvlan

pause-autoneg

Whether to automatically negotiate on pause frame of flow control mechanism defined by IEEE 802.3x standard.

pause-rx

Whether RX pause should be enabled. Only valid when automatic negotiation is disabled

pause-tx

Whether TX pause should be enabled. Only valid when automatic negotiation is disabled

ring-rx**ring-rx-jumbo****ring-rx-mini****ring-tx****gsm setting**

GSM-based Mobile Broadband Settings.

Properties:

apn

Alias: apn

The GPRS Access Point Name specifying the APN used when establishing a data session with the GSM-based network. The APN often determines how the user will be billed for their network usage and whether the user has access to the Internet or just a provider-specific walled-garden, so it is important to use the correct APN for the user's mobile broadband plan. The APN may only be composed of the characters a-z, 0-9, ., and – per GSM 03.60 Section 14.9.

Format: string

auto-config

When TRUE, the settings such as APN, username, or password will default to values that match the network the modem will register to in the Mobile Broadband Provider database.

Format: boolean

device-id

The device unique identifier (as given by the WWAN management service) which this connection applies to. If given, the connection will only apply to the specified device.

Format: string

home-only

When TRUE, only connections to the home network will be allowed. Connections to roaming networks will not be made.

Format: boolean

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple frames.

Format: uint32

network-id

The Network ID (GSM LAI format, ie MCC–MNC) to force specific network registration. If the Network ID is specified, NetworkManager will attempt to force the device to register only on the specified network. This can be used to ensure that the device does not roam when direct roaming control of the device is not otherwise possible.

Format: string

number

Legacy setting that used to help establishing PPP data sessions for GSM-based modems. Deprecated:
1

Format: string

password

Alias: password

The password used to authenticate with the network, if required. Many providers do not require a password, or accept any password. But if a password is required, it is specified here.

Format: string

password-flags

Flags indicating how to handle the "password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

pin

If the SIM is locked with a PIN it must be unlocked before any other operations are requested. Specify the PIN here to allow operation of the device.

Format: string

pin-flags

Flags indicating how to handle the "pin" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

sim-id

The SIM card unique identifier (as given by the WWAN management service) which this connection applies to. If given, the connection will apply to any device also allowed by "device-id" which contains a SIM card matching the given identifier.

Format: string

sim-operator-id

A MCC/MNC string like "310260" or "21601" identifying the specific mobile network operator which this connection applies to. If given, the connection will apply to any device also allowed by "device-id" and "sim-id" which contains a SIM card provisioned by the given operator.

Format: string

username

Alias: user

The username used to authenticate with the network, if required. Many providers do not require a username, or accept any username. But if a username is required, it is specified here.

Format: string

infiniband setting

Infiniband Settings.

Properties:

mac-address

Alias: mac

If specified, this connection will only apply to the IPoIB device whose permanent MAC address matches. This property does not change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple frames.

Format: uint32

p-key

Alias: p-key

The InfiniBand P_Key to use for this device. A value of -1 means to use the default P_Key (aka "the P_Key at index 0"). Otherwise, it is a 16-bit unsigned integer, whose high bit is set if it is a "full membership" P_Key.

Format: int32

parent

Alias: parent

The interface name of the parent device of this device. Normally NULL, but if the "p_key" property is set, then you must specify the base device by setting either this property or "mac-address".

Format: string

transport-mode

Alias: transport-mode

The IP-over-InfiniBand transport mode. Either "datagram" or "connected".

Format: string

ipv4 setting

IPv4 Settings.

Properties:

addresses

Alias: ip4

A list of IPv4 addresses and their prefix length. Multiple addresses can be separated by comma. For example "192.168.1.5/24, 10.1.0.5/24". The addresses are listed in decreasing priority, meaning the first address will be the primary address.

Format: a comma separated list of addresses

dad-timeout

Timeout in milliseconds used to check for the presence of duplicate IP addresses on the network. If an address conflict is detected, the activation will fail. A zero value means that no duplicate address detection is performed, -1 means the default value (either configuration ipvx.dad-timeout override or zero). A value greater than zero is a timeout in milliseconds. The property is currently implemented only for IPv4.

Format: int32

dhcp-client-id

A string sent to the DHCP server to identify the local machine which the DHCP server may use to customize the DHCP lease and options. When the property is a hex string ('aa:bb:cc') it is interpreted as a binary client ID, in which case the first byte is assumed to be the 'type' field as per RFC 2132 section 9.14 and the remaining bytes may be an hardware address (e.g. '01:xx:xx:xx:xx:xx' where 1 is the Ethernet ARP type and the rest is a MAC address). If the property is not a hex string it is considered as a non-hardware-address client ID and the 'type' field is set to 0. The special values "mac" and "perm-mac" are supported, which use the current or permanent MAC address of the device to generate a client identifier with type ethernet (01). Currently, these options only work for ethernet type of links. The special value "ipv6-duid" uses the DUID from "ipv6.dhcp-duid" property as an RFC4361-compliant client identifier. As IAID it uses "ipv4.dhcp-iaid" and falls back to "ipv6.dhcp-iaid" if unset. The special value "duid" generates a RFC4361-compliant client identifier based on "ipv4.dhcp-iaid" and uses a DUID generated by hashing /etc/machine-id. The special value "stable" is supported to generate a type 0 client identifier based on the stable-id (see connection.stable-id) and a per-host key. If you set the stable-id, you may want to include the "\${DEVICE}" or "\${MAC}" specifier to get a per-device key. If unset, a globally configured default is used. If still unset, the default depends on the DHCP plugin.

Format: string

dhcp-fqdn

If the "dhcp-send-hostname" property is TRUE, then the specified FQDN will be sent to the DHCP server when acquiring a lease. This property and "dhcp-hostname" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp-hostname

If the "dhcp-send-hostname" property is TRUE, then the specified name will be sent to the DHCP server when acquiring a lease. This property and "dhcp-fqdn" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp-hostname-flags

Flags for the DHCP hostname and FQDN. Currently, this property only includes flags to control the FQDN flags set in the DHCP FQDN option. Supported FQDN flags are NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) and NM_DHCP_HOSTNAME_FLAG_FQDN_NO_UPDATE (0x4). When no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is set, the DHCP FQDN option will contain no flag. Otherwise, if no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is not set, the standard FQDN flags are set in the request: NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) for IPv4 and NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1) for IPv6. When this property is set to the default value NM_DHCP_HOSTNAME_FLAG_NONE (0x0), a global default is looked up in NetworkManager configuration. If that value is unset or also NM_DHCP_HOSTNAME_FLAG_NONE (0x0), then the standard FQDN flags described above are sent in the DHCP requests.

Format: uint32

dhcp-iaid

A string containing the "Identity Association Identifier" (IAID) used by the DHCP client. The property is a 32-bit decimal value or a special value among "mac", "perm-mac", "ifname" and "stable". When set to "mac" (or "perm-mac"), the last 4 bytes of the current (or permanent) MAC address are used as IAIID. When set to "ifname", the IAIID is computed by hashing the interface name. The special value "stable" can be used to generate an IAIID based on the stable-id (see connection.stable-id), a per-host key and the interface name. When the property is unset, the value from global configuration is used; if no global default is set then the IAIID is assumed to be "ifname". Note that at the moment this property is ignored for IPv6 by dhclient, which always derives the IAIID from the MAC address.

Format: string

dhcp-reject-servers

Array of servers from which DHCP offers must be rejected. This property is useful to avoid getting a lease from misconfigured or rogue servers. For DHCPv4, each element must be an IPv4 address, optionally followed by a slash and a prefix length (e.g. "192.168.122.0/24"). This property is currently not implemented for DHCPv6.

Format: array of string

dhcp-send-hostname

If TRUE, a hostname is sent to the DHCP server when acquiring a lease. Some DHCP servers use this hostname to update DNS databases, essentially providing a static hostname for the computer. If the "dhcp-hostname" property is NULL and this property is TRUE, the current persistent hostname of the computer is sent.

Format: boolean

dhcp-timeout

A timeout for a DHCP transaction in seconds. If zero (the default), a globally configured default is used. If still unspecified, a device specific timeout is used (usually 45 seconds). Set to 2147483647

(MAXINT32) for infinity.

Format: int32

dhcp-vendor-class-identifier

The Vendor Class Identifier DHCP option (60). Special characters in the data string may be escaped using C-style escapes, nevertheless this property cannot contain nul bytes. If the per-profile value is unspecified (the default), a global connection default gets consulted. If still unspecified, the DHCP option is not sent to the server. Since 1.28

Format: string

dns

Array of IP addresses of DNS servers.

Format: array of uint32

dns-options

Array of DNS options as described in man 5 resolv.conf. NULL means that the options are unset and left at the default. In this case NetworkManager will use default options. This is distinct from an empty list of properties. The currently supported options are "attempts", "debug", "edns0", "inet6", "ip6-bytestring", "ip6-dotint", "ndots", "no-check-names", "no-ip6-dotint", "no-reload", "no-tld-query", "rotate", "single-request", "single-request-reopen", "timeout", "trust-ad", "use-vc". The "trust-ad" setting is only honored if the profile contributes name servers to resolv.conf, and if all contributing profiles have "trust-ad" enabled. When using a caching DNS plugin (dnsmasq or systemd-resolved in NetworkManager.conf) then "edns0" and "trust-ad" are automatically added.

Format: array of string

dns-priority

DNS servers priority. The relative priority for DNS servers specified by this setting. A lower numerical value is better (higher priority). Negative values have the special effect of excluding other configurations with a greater numerical priority value; so in presence of at least one negative priority, only DNS servers from connections with the lowest priority value will be used. To avoid all DNS leaks, set the priority of the profile that should be used to the most negative value of all active connections profiles. Zero selects a globally configured default value. If the latter is missing or zero too, it defaults to 50 for VPNs (including WireGuard) and 100 for other connections. Note that the priority is to order DNS settings for multiple active connections. It does not disambiguate multiple DNS servers within the same connection profile. When multiple devices have configurations with the same priority, VPNs will be considered first, then devices with the best (lowest metric) default route and then all other devices. When using dns=default, servers with higher priority will be on top of resolv.conf. To prioritize a given server over another one within the same connection, just specify them in the desired order. Note that commonly the resolver tries name servers in /etc/resolv.conf in the order listed, proceeding with the next server in the list on failure. See for example the "rotate" option of the dns-options setting. If there are any negative DNS priorities, then only name servers from the devices with that lowest priority will be considered. When using a DNS resolver that supports Conditional Forwarding or Split DNS (with dns=dnsmasq or dns=systemd-resolved settings), each connection is used to query domains in its search list. The search domains determine which name servers to ask, and the DNS priority is used to prioritize name servers based on the domain. Queries for domains not present in any search list are routed through connections having the `.' special wildcard domain, which is added automatically to connections with the default route (or can be added manually). When multiple connections specify the same domain, the one with the best priority (lowest numerical value) wins. If a sub domain is configured on another interface it will be accepted regardless the priority, unless parent domain on the other interface has a negative priority, which causes the sub domain to be shadowed. With Split DNS one can avoid undesired DNS leaks by properly configuring DNS priorities and the search domains, so that only name servers of the desired interface are configured.

Format: int32

dns-search

Array of DNS search domains. Domains starting with a tilde ('~') are considered 'routing' domains and are used only to decide the interface over which a query must be forwarded; they are not used to complete unqualified host names. When using a DNS plugin that supports Conditional Forwarding or Split DNS, then the search domains specify which name servers to query. This makes the behavior different from running with plain /etc/resolv.conf. For more information see also the dns-priority setting.

Format: array of string

gateway

Alias: gw4

The gateway associated with this configuration. This is only meaningful if "addresses" is also set. The gateway's main purpose is to control the next hop of the standard default route on the device. Hence, the gateway property conflicts with "never-default" and will be automatically dropped if the IP configuration is set to never-default. As an alternative to set the gateway, configure a static default route with /0 as prefix length.

Format: string

ignore-auto-dns

When "method" is set to "auto" and this property to TRUE, automatically configured name servers and search domains are ignored and only name servers and search domains specified in the "dns" and "dns-search" properties, if any, are used.

Format: boolean

ignore-auto-routes

When "method" is set to "auto" and this property to TRUE, automatically configured routes are ignored and only routes specified in the "routes" property, if any, are used.

Format: boolean

may-fail

If TRUE, allow overall network configuration to proceed even if the configuration specified by this property times out. Note that at least one IP configuration must succeed or overall network configuration will still fail. For example, in IPv6-only networks, setting this property to TRUE on the NMSettingIP4Config allows the overall network configuration to succeed if IPv4 configuration fails but IPv6 configuration completes successfully.

Format: boolean

method

IP configuration method. NMSettingIP4Config and NMSettingIP6Config both support "disabled", "auto", "manual", and "link-local". See the subclass-specific documentation for other values. In general, for the "auto" method, properties such as "dns" and "routes" specify information that is added on to the information returned from automatic configuration. The "ignore-auto-routes" and "ignore-auto-dns" properties modify this behavior. For methods that imply no upstream network, such as "shared" or "link-local", these properties must be empty. For IPv4 method "shared", the IP subnet can be configured by adding one manual IPv4 address or otherwise 10.42.x.0/24 is chosen. Note that the shared method must be configured on the interface which shares the internet to a subnet, not on the uplink which is shared.

Format: string

never-default

If TRUE, this connection will never be the default connection for this IP type, meaning it will never be assigned the default route by NetworkManager.

Format: boolean

required-timeout

The minimum time interval in milliseconds for which dynamic IP configuration should be tried before the connection succeeds. This property is useful for example if both IPv4 and IPv6 are enabled and are allowed to fail. Normally the connection succeeds as soon as one of the two address families completes; by setting a required timeout for e.g. IPv4, one can ensure that even if IP6 succeeds earlier than IPv4, NetworkManager waits some time for IPv4 before the connection becomes active. Note that if "may-fail" is FALSE for the same address family, this property has no effect as NetworkManager needs to wait for the full DHCP timeout. A zero value means that no required timeout is present, -1 means the default value (either configuration `ipvx.required-timeout` override or zero).

Format: int32

route-metric

The default metric for routes that don't explicitly specify a metric. The default value -1 means that the metric is chosen automatically based on the device type. The metric applies to dynamic routes, manual (static) routes that don't have an explicit metric setting, address prefix routes, and the default route. Note that for IPv6, the kernel accepts zero (0) but coerces it to 1024 (user default). Hence, setting this property to zero effectively mean setting it to 1024. For IPv4, zero is a regular value for the metric.

Format: int64

route-table

Enable policy routing (source routing) and set the routing table used when adding routes. This affects all routes, including device-routes, IPv4LL, DHCP, SLAAC, default-routes and static routes. But note that static routes can individually overwrite the setting by explicitly specifying a non-zero routing table. If the table setting is left at zero, it is eligible to be overwritten via global configuration. If the property is zero even after applying the global configuration value, policy routing is disabled for the address family of this connection. Policy routing disabled means that NetworkManager will add all routes to the main table (except static routes that explicitly configure a different table). Additionally, NetworkManager will not delete any extraneous routes from tables except the main table. This is to preserve backward compatibility for users who manage routing tables outside of NetworkManager.

Format: uint32

routes

A list of IPv4 destination addresses, prefix length, optional IPv4 next hop addresses, optional route metric, optional attribute. The valid syntax is: "ip[/prefix] [next-hop] [metric] [attribute=val]...[,ip[/prefix]...]". For example "192.0.2.0/24 10.1.1.1 77, 198.51.100.0/24".

Various attributes are supported:

- "cwnd" – an unsigned 32 bit integer.
- "initcwnd" – an unsigned 32 bit integer.
- "inirwnd" – an unsigned 32 bit integer.
- "lock-cwnd" – a boolean value.
- "lock-initcwnd" – a boolean value.
- "lock-inirwnd" – a boolean value.
- "lock-mtu" – a boolean value.
- "lock-window" – a boolean value.

- "mtu" – an unsigned 32 bit integer.
- "onlink" – a boolean value.
- "scope" – an unsigned 8 bit integer. IPv4 only.
- "src" – an IPv4 address.
- "table" – an unsigned 32 bit integer. The default depends on ipv4.route-table.
- "tos" – an unsigned 8 bit integer. IPv4 only.
- "type" – one of unicast, local, blackhole, unavailable, prohibit. The default is unicast.
- "window" – an unsigned 32 bit integer.

For details see also ‘man ip–route‘.

Format: a comma separated list of routes

routing–rules

A comma separated list of routing rules for policy routing. The format is based on **ip rule add** syntax and mostly compatible. One difference is that routing rules in NetworkManager always need a fixed priority.

Example: priority 5 from 192.167.4.0/24 table 45

Format: a comma separated list of routing rules

ipv6 setting

IPv6 Settings.

Properties:

addr–gen–mode

Configure method for creating the address for use with RFC4862 IPv6 Stateless Address

Autoconfiguration. The permitted values are:

NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_EUI64 (0) or

NM_SETTING_IP6_CONFIG_ADDR_GEN_MODE_STABLE_PRIVACY (1). If the property is set to EUI64, the addresses will be generated using the interface tokens derived from hardware address.

This makes the host part of the address to stay constant, making it possible to track host's presence when it changes networks. The address changes when the interface hardware is replaced. The value of stable–privacy enables use of cryptographically secure hash of a secret host–specific key along with the connection's stable–id and the network address as specified by RFC7217. This makes it impossible to use the address track host's presence, and makes the address stable when the network interface hardware is replaced. On D-Bus, the absence of an addr–gen–mode setting equals enabling stable–privacy. For keyfile plugin, the absence of the setting on disk means EUI64 so that the property doesn't change on upgrade from older versions. Note that this setting is distinct from the Privacy Extensions as configured by "ip6–privacy" property and it does not affect the temporary addresses configured with this option.

Format: int32

addresses

Alias: ip6

A list of IPv6 addresses and their prefix length. Multiple addresses can be separated by comma. For example "2001:db8:85a3::8a2e:370:7334/64, 2001:db8:85a3::5/64". The addresses are listed in increasing priority, meaning the last address will be the primary address.

Format: a comma separated list of addresses

dhcp-duid

A string containing the DHCPv6 Unique Identifier (DUID) used by the dhcp client to identify itself to DHCPv6 servers (RFC 3315). The DUID is carried in the Client Identifier option. If the property is a hex string ('aa:bb:cc') it is interpreted as a binary DUID and filled as an opaque value in the Client Identifier option. The special value "lease" will retrieve the DUID previously used from the lease file belonging to the connection. If no DUID is found and "dhclient" is the configured dhcp client, the DUID is searched in the system-wide dhclient lease file. If still no DUID is found, or another dhcp client is used, a global and permanent DUID–UUID (RFC 6355) will be generated based on the machine–id. The special values "llt" and "ll" will generate a DUID of type LLT or LL (see RFC 3315) based on the current MAC address of the device. In order to try providing a stable DUID–LLT, the time field will contain a constant timestamp that is used globally (for all profiles) and persisted to disk. The special values "stable–llt", "stable–ll" and "stable–uuid" will generate a DUID of the corresponding type, derived from the connection's stable–id and a per–host unique key. You may want to include the "\${DEVICE}" or "\${MAC}" specifier in the stable–id, in case this profile gets activated on multiple devices. So, the link–layer address of "stable–ll" and "stable–llt" will be a generated address derived from the stable id. The DUID–LLT time value in the "stable–llt" option will be picked among a static timespan of three years (the upper bound of the interval is the same constant timestamp used in "llt"). When the property is unset, the global value provided for "ipv6.dhcp–duid" is used. If no global value is provided, the default "lease" value is assumed.

Format: string

dhcp–hostname

If the "dhcp–send–hostname" property is TRUE, then the specified name will be sent to the DHCP server when acquiring a lease. This property and "dhcp–fqdn" are mutually exclusive and cannot be set at the same time.

Format: string

dhcp–hostname–flags

Flags for the DHCP hostname and FQDN. Currently, this property only includes flags to control the FQDN flags set in the DHCP FQDN option. Supported FQDN flags are NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) and NM_DHCP_HOSTNAME_FLAG_FQDN_NO_UPDATE (0x4). When no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is set, the DHCP FQDN option will contain no flag. Otherwise, if no FQDN flag is set and NM_DHCP_HOSTNAME_FLAG_FQDN_CLEAR_FLAGS (0x8) is not set, the standard FQDN flags are set in the request: NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1), NM_DHCP_HOSTNAME_FLAG_FQDN_ENCODED (0x2) for IPv4 and NM_DHCP_HOSTNAME_FLAG_FQDN_SERV_UPDATE (0x1) for IPv6. When this property is set to the default value NM_DHCP_HOSTNAME_FLAG_NONE (0x0), a global default is looked up in NetworkManager configuration. If that value is unset or also NM_DHCP_HOSTNAME_FLAG_NONE (0x0), then the standard FQDN flags described above are sent in the DHCP requests.

Format: uint32

dhcp–iaid

A string containing the "Identity Association Identifier" (IAID) used by the DHCP client. The property is a 32-bit decimal value or a special value among "mac", "perm–mac", "ifname" and "stable". When set to "mac" (or "perm–mac"), the last 4 bytes of the current (or permanent) MAC address are used as IAIID. When set to "ifname", the IAIID is computed by hashing the interface name. The special value "stable" can be used to generate an IAIID based on the stable–id (see connection.stable–id), a per–host key and the interface name. When the property is unset, the value from global configuration is used; if no global default is set then the IAIID is assumed to be "ifname". Note that at the moment this property

is ignored for IPv6 by dhclient, which always derives the IAID from the MAC address.

Format: string

dhcp-send-hostname

If TRUE, a hostname is sent to the DHCP server when acquiring a lease. Some DHCP servers use this hostname to update DNS databases, essentially providing a static hostname for the computer. If the "dhcp-hostname" property is NULL and this property is TRUE, the current persistent hostname of the computer is sent.

Format: boolean

dhcp-timeout

A timeout for a DHCP transaction in seconds. If zero (the default), a globally configured default is used. If still unspecified, a device specific timeout is used (usually 45 seconds). Set to 2147483647 (MAXINT32) for infinity.

Format: int32

dns

Array of IP addresses of DNS servers.

Format: array of byte array

dns-options

Array of DNS options as described in man 5 resolv.conf. NULL means that the options are unset and left at the default. In this case NetworkManager will use default options. This is distinct from an empty list of properties. The currently supported options are "attempts", "debug", "edns0", "inet6", "ip6-bytes", "ip6-dotint", "ndots", "no-check-names", "no-ip6-dotint", "no-reload", "no-tld-query", "rotate", "single-request", "single-request-reopen", "timeout", "trust-ad", "use-vc". The "trust-ad" setting is only honored if the profile contributes name servers to resolv.conf, and if all contributing profiles have "trust-ad" enabled. When using a caching DNS plugin (dnsmasq or systemd-resolved in NetworkManager.conf) then "edns0" and "trust-ad" are automatically added.

Format: array of string

dns-priority

DNS servers priority. The relative priority for DNS servers specified by this setting. A lower numerical value is better (higher priority). Negative values have the special effect of excluding other configurations with a greater numerical priority value; so in presence of at least one negative priority, only DNS servers from connections with the lowest priority value will be used. To avoid all DNS leaks, set the priority of the profile that should be used to the most negative value of all active connections profiles. Zero selects a globally configured default value. If the latter is missing or zero too, it defaults to 50 for VPNs (including WireGuard) and 100 for other connections. Note that the priority is to order DNS settings for multiple active connections. It does not disambiguate multiple DNS servers within the same connection profile. When multiple devices have configurations with the same priority, VPNs will be considered first, then devices with the best (lowest metric) default route and then all other devices. When using dns=default, servers with higher priority will be on top of resolv.conf. To prioritize a given server over another one within the same connection, just specify them in the desired order. Note that commonly the resolver tries name servers in /etc/resolv.conf in the order listed, proceeding with the next server in the list on failure. See for example the "rotate" option of the dns-options setting. If there are any negative DNS priorities, then only name servers from the devices with that lowest priority will be considered. When using a DNS resolver that supports Conditional Forwarding or Split DNS (with dns=dnsmasq or dns=systemd-resolved settings), each connection is used to query domains in its search list. The search domains determine which name servers to ask, and the DNS priority is used to prioritize name servers based on the domain. Queries for domains not present in any search list are routed through connections having the '^' special

wildcard domain, which is added automatically to connections with the default route (or can be added manually). When multiple connections specify the same domain, the one with the best priority (lowest numerical value) wins. If a sub domain is configured on another interface it will be accepted regardless the priority, unless parent domain on the other interface has a negative priority, which causes the sub domain to be shadowed. With Split DNS one can avoid undesired DNS leaks by properly configuring DNS priorities and the search domains, so that only name servers of the desired interface are configured.

Format: int32

dns-search

Array of DNS search domains. Domains starting with a tilde ('~') are considered 'routing' domains and are used only to decide the interface over which a query must be forwarded; they are not used to complete unqualified host names. When using a DNS plugin that supports Conditional Forwarding or Split DNS, then the search domains specify which name servers to query. This makes the behavior different from running with plain /etc/resolv.conf. For more information see also the dns-priority setting.

Format: array of string

gateway

Alias: gw6

The gateway associated with this configuration. This is only meaningful if "addresses" is also set. The gateway's main purpose is to control the next hop of the standard default route on the device. Hence, the gateway property conflicts with "never-default" and will be automatically dropped if the IP configuration is set to never-default. As an alternative to set the gateway, configure a static default route with /0 as prefix length.

Format: string

ignore-auto-dns

When "method" is set to "auto" and this property to TRUE, automatically configured name servers and search domains are ignored and only name servers and search domains specified in the "dns" and "dns-search" properties, if any, are used.

Format: boolean

ignore-auto-routes

When "method" is set to "auto" and this property to TRUE, automatically configured routes are ignored and only routes specified in the "routes" property, if any, are used.

Format: boolean

ip6-privacy

Configure IPv6 Privacy Extensions for SLAAC, described in RFC4941. If enabled, it makes the kernel generate a temporary IPv6 address in addition to the public one generated from MAC address via modified EUI-64. This enhances privacy, but could cause problems in some applications, on the other hand. The permitted values are: -1: unknown, 0: disabled, 1: enabled (prefer public address), 2: enabled (prefer temporary addresses). Having a per-connection setting set to "-1" (unknown) means fallback to global configuration "ipv6.ip6-privacy". If also global configuration is unspecified or set to "-1", fallback to read "/proc/sys/net/ipv6/conf/default/use_tempaddr". Note that this setting is distinct from the Stable Privacy addresses that can be enabled with the "addr-gen-mode" property's "stable-privacy" setting as another way of avoiding host tracking with IPv6 addresses.

Format: NMSettingIP6ConfigPrivacy (int32)

may-fail

If TRUE, allow overall network configuration to proceed even if the configuration specified by this property times out. Note that at least one IP configuration must succeed or overall network configuration will still fail. For example, in IPv6-only networks, setting this property to TRUE on the NMSettingIP4Config allows the overall network configuration to succeed if IPv4 configuration fails but IPv6 configuration completes successfully.

Format: boolean

method

IP configuration method. NMSettingIP4Config and NMSettingIP6Config both support "disabled", "auto", "manual", and "link-local". See the subclass-specific documentation for other values. In general, for the "auto" method, properties such as "dns" and "routes" specify information that is added on to the information returned from automatic configuration. The "ignore-auto-routes" and "ignore-auto-dns" properties modify this behavior. For methods that imply no upstream network, such as "shared" or "link-local", these properties must be empty. For IPv4 method "shared", the IP subnet can be configured by adding one manual IPv4 address or otherwise 10.42.x.0/24 is chosen. Note that the shared method must be configured on the interface which shares the internet to a subnet, not on the uplink which is shared.

Format: string

never-default

If TRUE, this connection will never be the default connection for this IP type, meaning it will never be assigned the default route by NetworkManager.

Format: boolean

ra-timeout

A timeout for waiting Router Advertisements in seconds. If zero (the default), a globally configured default is used. If still unspecified, the timeout depends on the sysctl settings of the device. Set to 2147483647 (MAXINT32) for infinity.

Format: int32

required-timeout

The minimum time interval in milliseconds for which dynamic IP configuration should be tried before the connection succeeds. This property is useful for example if both IPv4 and IPv6 are enabled and are allowed to fail. Normally the connection succeeds as soon as one of the two address families completes; by setting a required timeout for e.g. IPv4, one can ensure that even if IP6 succeeds earlier than IPv4, NetworkManager waits some time for IPv4 before the connection becomes active. Note that if "may-fail" is FALSE for the same address family, this property has no effect as NetworkManager needs to wait for the full DHCP timeout. A zero value means that no required timeout is present, -1 means the default value (either configuration ipvx.required-timeout override or zero).

Format: int32

route-metric

The default metric for routes that don't explicitly specify a metric. The default value -1 means that the metric is chosen automatically based on the device type. The metric applies to dynamic routes, manual (static) routes that don't have an explicit metric setting, address prefix routes, and the default route. Note that for IPv6, the kernel accepts zero (0) but coerces it to 1024 (user default). Hence, setting this property to zero effectively mean setting it to 1024. For IPv4, zero is a regular value for the metric.

Format: int64

route-table

Enable policy routing (source routing) and set the routing table used when adding routes. This affects all routes, including device-routes, IPv4LL, DHCP, SLAAC, default-routes and static routes. But note

that static routes can individually overwrite the setting by explicitly specifying a non-zero routing table. If the table setting is left at zero, it is eligible to be overwritten via global configuration. If the property is zero even after applying the global configuration value, policy routing is disabled for the address family of this connection. Policy routing disabled means that NetworkManager will add all routes to the main table (except static routes that explicitly configure a different table). Additionally, NetworkManager will not delete any extraneous routes from tables except the main table. This is to preserve backward compatibility for users who manage routing tables outside of NetworkManager.

Format: uint32

routes

A list of IPv6 destination addresses, prefix length, optional IPv6 next hop addresses, optional route metric, optional attribute. The valid syntax is: "ip[/prefix] [next-hop] [metric] [attribute=val]...[,ip[/prefix]...]".

Various attributes are supported:

- "cwnd" – an unsigned 32 bit integer.
- "from" – an IPv6 address with optional prefix. IPv6 only.
- "initcwnd" – an unsigned 32 bit integer.
- "initrwnd" – an unsigned 32 bit integer.
- "lock-cwnd" – a boolean value.
- "lock-initcwnd" – a boolean value.
- "lock-initrwnd" – a boolean value.
- "lock-mtu" – a boolean value.
- "lock-window" – a boolean value.
- "mtu" – an unsigned 32 bit integer.
- "onlink" – a boolean value.
- "src" – an IPv6 address.
- "table" – an unsigned 32 bit integer. The default depends on ipv6.route-table.
- "type" – one of unicast, local, blackhole, unavailable, prohibit. The default is unicast.
- "window" – an unsigned 32 bit integer.

For details see also ‘man ip-route’.

Format: a comma separated list of routes

routing-rules

A comma separated list of routing rules for policy routing. The format is based on **ip rule add** syntax and mostly compatible. One difference is that routing rules in NetworkManager always need a fixed priority.

Example: priority 5 from 1:2:3::5/128 table 45

Format: a comma separated list of routing rules

token

Configure the token for draft-chown-6man-tokenised-ipv6-identifiers-02 IPv6 tokenized interface identifiers. Useful with eui64 addr-gen-mode.

Format: string

ip-tunnel setting

IP Tunneling Settings.

Properties:

encapsulation-limit

How many additional levels of encapsulation are permitted to be prepended to packets. This property applies only to IPv6 tunnels.

Format: uint32

flags

Tunnel flags. Currently, the following values are supported:

NM_IP_TUNNEL_FLAG_IP6_IGN_ENCAP_LIMIT (0x1),

NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_TCLASS (0x2),

NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_FLOWLABEL (0x4),

NM_IP_TUNNEL_FLAG_IP6_MIP6_DEV (0x8),

NM_IP_TUNNEL_FLAG_IP6_RCV_DSCP_COPY (0x10),

NM_IP_TUNNEL_FLAG_IP6_USE_ORIG_FWMARK (0x20). They are valid only for IPv6 tunnels.

Format: uint32

flow-label

The flow label to assign to tunnel packets. This property applies only to IPv6 tunnels.

Format: uint32

input-key

The key used for tunnel input packets; the property is valid only for certain tunnel modes (GRE, IP6GRE). If empty, no key is used.

Format: string

local

Alias: local

The local endpoint of the tunnel; the value can be empty, otherwise it must contain an IPv4 or IPv6 address.

Format: string

mode

Alias: mode

The tunneling mode, for example NM_IP_TUNNEL_MODE_IPIP (1) or NM_IP_TUNNEL_MODE_GRE (2).

Format: uint32

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple fragments.

Format: uint32

output-key

The key used for tunnel output packets; the property is valid only for certain tunnel modes (GRE, IP6GRE). If empty, no key is used.

Format: string

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID the new device will be bound to so that tunneled packets will only be routed via that interface.

Format: string

path-mtu-discovery

Whether to enable Path MTU Discovery on this tunnel.

Format: boolean

remote

Alias: remote

The remote endpoint of the tunnel; the value must contain an IPv4 or IPv6 address.

Format: string

tos

The type of service (IPv4) or traffic class (IPv6) field to be set on tunneled packets.

Format: uint32

ttl

The TTL to assign to tunneled packets. 0 is a special value meaning that packets inherit the TTL value.

Format: uint32

macsec setting

MACSec Settings.

Properties:

encrypt

Alias: encrypt

Whether the transmitted traffic must be encrypted.

Format: boolean

mka-cak

Alias: cak

The pre-shared CAK (Connectivity Association Key) for MACsec Key Agreement.

Format: string

mka-cak-flags

Flags indicating how to handle the "mka-cak" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

mka-ckn

Alias: ckn

The pre-shared CKN (Connectivity-association Key Name) for MACsec Key Agreement.

Format: string

mode

Alias: mode

Specifies how the CAK (Connectivity Association Key) for MKA (MACsec Key Agreement) is obtained.

Format: int32

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this MACSEC interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

port

Alias: port

The port component of the SCI (Secure Channel Identifier), between 1 and 65534.

Format: int32

send-sci

Specifies whether the SCI (Secure Channel Identifier) is included in every packet.

Format: boolean

validation

Specifies the validation mode for incoming frames.

Format: int32

macvlan setting

MAC VLAN Settings.

Properties:

mode

Alias: mode

The macvlan mode, which specifies the communication mechanism between multiple macvlans on the same lower device.

Format: uint32

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this MAC-VLAN interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

promiscuous

Whether the interface should be put in promiscuous mode.

Format: boolean

tap

Alias: tap

Whether the interface should be a MACVTAP.

Format: boolean

match setting

Match settings.

Properties:

driver

A list of driver names to match. Each element is a shell wildcard pattern. See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\' are used for optional and mandatory matches and inverting the pattern.

Format: array of string

interface-name

A list of interface names to match. Each element is a shell wildcard pattern. An element can be prefixed with a pipe symbol (|) or an ampersand (&). The former means that the element is optional and the latter means that it is mandatory. If there are any optional elements, than the match evaluates to true if at least one of the optional element matches (logical OR). If there are any mandatory elements, then they all must match (logical AND). By default, an element is optional. This means that an element "foo" behaves the same as "|foo". An element can also be inverted with exclamation mark (!) between the pipe symbol (or the ampersand) and before the pattern. Note that "!foo" is a shortcut for the mandatory match "&!foo". Finally, a backslash can be used at the beginning of the element (after the optional special characters) to escape the start of the pattern. For example, "&\!a" is an mandatory match for literally "\a".

Format: array of string

kernel-command-line

A list of kernel command line arguments to match. This may be used to check whether a specific kernel command line option is set (or unset, if prefixed with the exclamation mark). The argument must either be a single word, or an assignment (i.e. two words, joined by "="). In the former case the kernel command line is searched for the word appearing as is, or as left hand side of an assignment. In the latter case, the exact assignment is looked for with right and left hand side matching. Wildcard patterns are not supported. See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\' are used for optional and mandatory matches and inverting the match.

Format: array of string

path

A list of paths to match against the ID_PATH udev property of devices. ID_PATH represents the topological persistent path of a device. It typically contains a subsystem string (pci, usb, platform, etc.) and a subsystem-specific identifier. For PCI devices the path has the form "pci-\$domain:\$bus:\$device.\$function", where each variable is an hexadecimal value; for example "pci-0000:0a:00.0". The path of a device can be obtained with "udevadm info /sys/class/net/\$dev | grep ID_PATH=" or by looking at the "path" property exported by NetworkManager ("nmcli -f general.path device show \$dev"). Each element of the list is a shell wildcard pattern. See NMSettingMatch:interface-name for how special characters '|', '&', '!' and '\' are used for optional and mandatory matches and inverting the pattern.

Format: array of string

802-11-olpc-mesh setting

Alias: olpc-mesh

OLPC Wireless Mesh Settings.

Properties:

channel

Alias: channel

Channel on which the mesh network to join is located.

Format: uint32

dhcp-anycast-address

Alias: dhcp-anycast

Anycast DHCP MAC address used when requesting an IP address via DHCP. The specific anycast address used determines which DHCP server class answers the request. This is currently only implemented by dhclient DHCP plugin.

Format: byte array

ssid

Alias: ssid

SSID of the mesh network to join.

Format: byte array

ovs-bridge setting

OvsBridge Link Settings.

Properties:

datapath-type

The data path type. One of "system", "netdev" or empty.

Format: string

fail-mode

The bridge failure mode. One of "secure", "standalone" or empty.

Format: string

mcast-snooping-enable

Enable or disable multicast snooping.

Format: boolean

rstp-enable

Enable or disable RSTP.

Format: boolean

stp-enable

Enable or disable STP.

Format: boolean

ovs-dpdk setting

OvsDpdk Link Settings.

Properties:

devargs

Open vSwitch DPDK device arguments.

Format: string

n-rxq

Open vSwitch DPDK number of rx queues. Defaults to zero which means to leave the parameter in OVS unspecified and effectively configures one queue.

Format: uint32

ovs-interface setting

Open vSwitch Interface Settings.

Properties:

type

The interface type. Either "internal", "system", "patch", "dpdk", or empty.

Format: string

ovs-patch setting

OvsPatch Link Settings.

Properties:

peer

Specifies the name of the interface for the other side of the patch. The patch on the other side must also set this interface as peer.

Format: string

ovs-port setting

OvsPort Link Settings.

Properties:

bond-downdelay

The time port must be inactive in order to be considered down.

Format: uint32

bond-mode

Bonding mode. One of "active-backup", "balance-slb", or "balance-tcp".

Format: string

bond-updelay

The time port must be active before it starts forwarding traffic.

Format: uint32

lacp

LACP mode. One of "active", "off", or "passive".

Format: string

tag

The VLAN tag in the range 0–4095.

Format: uint32

vlan-mode

The VLAN mode. One of "access", "native-tagged", "native-untagged", "trunk" or unset.

Format: string

ppp setting

Point-to-Point Protocol Settings.

Properties:

baud

If non-zero, instruct pppd to set the serial port to the specified baudrate. This value should normally be left as 0 to automatically choose the speed.

Format: uint32

crtsets

If TRUE, specify that pppd should set the serial port to use hardware flow control with RTS and CTS signals. This value should normally be set to FALSE.

Format: boolean

lcp-echo-failure

If non-zero, instruct pppd to presume the connection to the peer has failed if the specified number of LCP echo-requests go unanswered by the peer. The "lcp-echo-interval" property must also be set to a non-zero value if this property is used.

Format: uint32

lcp-echo-interval

If non-zero, instruct pppd to send an LCP echo-request frame to the peer every n seconds (where n is the specified value). Note that some PPP peers will respond to echo requests and some will not, and it is not possible to autodetect this.

Format: uint32

mppe-stateful

If TRUE, stateful MPPE is used. See pppd documentation for more information on stateful MPPE.

Format: boolean

mru

If non-zero, instruct pppd to request that the peer send packets no larger than the specified size. If non-zero, the MRU should be between 128 and 16384.

Format: uint32

mtu

If non-zero, instruct pppd to send packets no larger than the specified size.

Format: uint32

no-vj-comp

If TRUE, Van Jacobson TCP header compression will not be requested.

Format: boolean

noauth

If TRUE, do not require the other side (usually the PPP server) to authenticate itself to the client. If FALSE, require authentication from the remote side. In almost all cases, this should be TRUE.

Format: boolean

nobsdcomp

If TRUE, BSD compression will not be requested.

Format: boolean

nodeflate

If TRUE, "deflate" compression will not be requested.

Format: boolean

refuse-chap

If TRUE, the CHAP authentication method will not be used.

Format: boolean

refuse-eap

If TRUE, the EAP authentication method will not be used.

Format: boolean

refuse-mschap

If TRUE, the MSCHAP authentication method will not be used.

Format: boolean

refuse-mschapv2

If TRUE, the MSCHAPv2 authentication method will not be used.

Format: boolean

refuse-pap

If TRUE, the PAP authentication method will not be used.

Format: boolean

require-mppe

If TRUE, MPPE (Microsoft Point-to-Point Encryption) will be required for the PPP session. If either 64-bit or 128-bit MPPE is not available the session will fail. Note that MPPE is not used on mobile broadband connections.

Format: boolean

require-mppe-128

If TRUE, 128-bit MPPE (Microsoft Point-to-Point Encryption) will be required for the PPP session, and the "require-mppe" property must also be set to TRUE. If 128-bit MPPE is not available the session will fail.

Format: boolean

pppoe setting

PPP-over-Ethernet Settings.

Properties:

parent

Alias: parent

If given, specifies the parent interface name on which this PPPoE connection should be created. If this property is not specified, the connection is activated on the interface specified in "interface-name" of NMSettingConnection.

Format: string

password

Alias: password

Password used to authenticate with the PPPoE service.

Format: string

password-flags

Flags indicating how to handle the "password" property. See the section called "Secret flag types:" for flag values.

Format: NMSettingSecretFlags (uint32)

service

Alias: service

If specified, instruct PPPoE to only initiate sessions with access concentrators that provide the specified service. For most providers, this should be left blank. It is only required if there are multiple access concentrators or a specific service is known to be required.

Format: string

username

Alias: username

Username used to authenticate with the PPPoE service.

Format: string

proxy setting

WWW Proxy Settings.

Properties:

browser-only

Alias: browser-only

Whether the proxy configuration is for browser only.

Format: boolean

method

Alias: method

Method for proxy configuration, Default is NM_SETTING_PROXY_METHOD_NONE (0)

Format: int32

pac-script

Alias: pac-script

PAC script for the connection.

Format: string

pac-url

Alias: pac-url

PAC URL for obtaining PAC file.

Format: string

serial setting

Serial Link Settings.

Properties:

baud

Speed to use for communication over the serial port. Note that this value usually has no effect for mobile broadband modems as they generally ignore speed settings and use the highest available speed.

Format: uint32

bits

Byte-width of the serial communication. The 8 in "8n1" for example.

Format: uint32

parity

Parity setting of the serial port.

Format: NMSettingSerialParity (byte)

send-delay

Time to delay between each byte sent to the modem, in microseconds.

Format: uint64

stopbits

Number of stop bits for communication on the serial port. Either 1 or 2. The 1 in "8n1" for example.

Format: uint32

sriov setting

SR-IOV settings.

Properties:

autoprobe-drivers

Whether to autoprobe virtual functions by a compatible driver. If set to NM_TERNARY_TRUE (1), the kernel will try to bind VFs to a compatible driver and if this succeeds a new network interface will be instantiated for each VF. If set to NM_TERNARY_FALSE (0), VFs will not be claimed and no network interfaces will be created for them. When set to NM_TERNARY_DEFAULT (-1), the global default is used; in case the global default is unspecified it is assumed to be NM_TERNARY_TRUE (1).

Format: NMternary (int32)

total-vfs

The total number of virtual functions to create. Note that when the sriov setting is present NetworkManager enforces the number of virtual functions on the interface (also when it is zero) during activation and resets it upon deactivation. To prevent any changes to SR-IOV parameters don't add a sriov setting to the connection.

Format: uint32

vfs

Array of virtual function descriptors. Each VF descriptor is a dictionary mapping attribute names to GVariant values. The 'index' entry is mandatory for each VF. When represented as string a VF is in the form: "INDEX [ATTR=VALUE[ATTR=VALUE]...]". for example: "2 mac=00:11:22:33:44:55

`spoof-check=true".` Multiple VFs can be specified using a comma as separator. Currently, the following attributes are supported: mac, spoof-check, trust, min-tx-rate, max-tx-rate, vlans. The "vlans" attribute is represented as a semicolon-separated list of VLAN descriptors, where each descriptor has the form "ID[.PRIORITY[.PROTO]]". PROTO can be either 'q' for 802.1Q (the default) or 'ad' for 802.1ad.

Format: array of vardict

tc setting

Linux Traffic Control Settings.

Properties:

qdiscs

Array of TC queueing disciplines. qdisc is a basic block in the Linux traffic control subsystem

Each qdisc can be specified by the following attributes:

handle HANDLE

specifies the qdisc handle. A qdisc, which potentially can have children, gets assigned a major number, called a 'handle', leaving the minor number namespace available for classes. The handle is expressed as '10:'. It is customary to explicitly assign a handle to qdiscs expected to have children.

parent HANDLE

specifies the handle of the parent qdisc the current qdisc must be attached to.

root

specifies that the qdisc is attached to the root of device.

KIND

this is the qdisc kind. NetworkManager currently supports the following kinds: fq_codel, sfq, tbf. Each qdisc kind has a different set of parameters, described below. There are also some kinds like pfifo, pfifo_fast, prio supported by NetworkManager but their parameters are not supported by NetworkManager.

Parameters for 'fq_codel':

limit U32

the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Default is 10240 packets.

memory_limit U32

sets a limit on the total number of bytes that can be queued in this FQ-CoDel instance. The lower of the packet limit of the limit parameter and the memory limit will be enforced. Default is 32 MB.

flows U32

the number of flows into which the incoming packets are classified. Due to the stochastic nature of hashing, multiple flows may end up being hashed into the same slot. Newer flows have priority over older ones. This parameter can be set only at load time since memory has to be allocated for the hash table. Default value is 1024.

target U32

the acceptable minimum standing/persistent queue delay. This minimum delay is identified by tracking the local minimum queue delay that packets experience. The unit of measurement is microsecond(us). Default value is 5ms.

interval U32

used to ensure that the measured minimum delay does not become too stale. The minimum delay must be experienced in the last epoch of length .B interval. It should be set on the order of the worst-case RTT through the bottleneck to give endpoints sufficient time to react. Default value is

100ms.

quantum U32

the number of bytes used as 'deficit' in the fair queuing algorithm. Default is set to 1514 bytes which corresponds to the Ethernet MTU plus the hardware header length of 14 bytes.

ecn BOOL

can be used to mark packets instead of dropping them. *ecn* is turned on by default.

ce_threshold U32

sets a threshold above which all packets are marked with ECN Congestion Experienced. This is useful for DCTCP-style congestion control algorithms that require marking at very shallow queueing thresholds.

Parameters for 'sfq':

divisor U32

can be used to set a different hash table size, available from kernel 2.6.39 onwards. The specified divisor must be a power of two and cannot be larger than 65536. Default value: 1024.

limit U32

Upper limit of the SFQ. Can be used to reduce the default length of 127 packets.

depth U32

Limit of packets per flow. Default to 127 and can be lowered.

perturb_period U32

Interval in seconds for queue algorithm perturbation. Defaults to 0, which means that no perturbation occurs. Do not set too low for each perturbation may cause some packet reordering or losses. Advised value: 60 This value has no effect when external flow classification is used. Its better to increase divisor value to lower risk of hash collisions.

quantum U32

Amount of bytes a flow is allowed to dequeue during a round of the round robin process. Defaults to the MTU of the interface which is also the advised value and the minimum value.

flows U32

Default value is 127.

Parameters for 'tbf':

rate U64

Bandwidth or rate. These parameters accept a floating point number, possibly followed by either a unit (both SI and IEC units supported), or a float followed by a percent character to specify the rate as a percentage of the device's speed.

burst U32

Also known as buffer or maxburst. Size of the bucket, in bytes. This is the maximum amount of bytes that tokens can be available for instantaneously. In general, larger shaping rates require a larger buffer. For 10mbit/s on Intel, you need at least 10kbyte buffer if you want to reach your configured rate!

If your buffer is too small, packets may be dropped because more tokens arrive per timer tick than fit in your bucket. The minimum buffer size can be calculated by dividing the rate by HZ.

Token usage calculations are performed using a table which by default has a resolution of 8 packets. This resolution can be changed by specifying the cell size with the burst. For example, to specify a 6000 byte buffer with a 16 byte cell size, set a burst of 6000/16. You will probably never have to set this. Must be an integral power of 2.

limit U32

Limit is the number of bytes that can be queued waiting for tokens to become available.

latency U32

specifies the maximum amount of time a packet can sit in the TBF. The latency calculation takes into account the size of the bucket, the rate and possibly the peakrate (if set). The latency and limit are mutually exclusive.

Format: GPtrArray(NMTCQdisc)

tfilters

Array of TC traffic filters. Traffic control can manage the packet content during classification by using filters.

Each tfilters can be specified by the following attributes:

handle HANDLE

specifies the tfilters handle. A filter is used by a classful qdisc to determine in which class a packet will be enqueued. It is important to notice that filters reside within qdiscs. Therefore, see qdiscs handle for detailed information.

parent HANDLE

specifies the handle of the parent qdisc the current qdisc must be attached to.

root

specifies that the qdisc is attached to the root of device.

KIND

this is the tfilters kind. NetworkManager currently supports following kinds: mirred, simple. Each filter kind has a different set of actions, described below. There are also some other kinds like matchall, basic, u32 supported by NetworkManager.

Actions for 'mirred':

egress bool

Define whether the packet should exit from the interface.

ingress bool

Define whether the packet should come into the interface.

mirror bool

Define whether the packet should be copied to the destination space.

redirect bool

Define whether the packet should be moved to the destination space.

Action for 'simple':

sdata char[32]

The actual string to print.

Format: GPtrArray(NMTCTfilter)

team setting

Teaming Settings.

Properties:

config

Alias: config

The JSON configuration for the team network interface. The property should contain raw JSON configuration data suitable for teamd, because the value is passed directly to teamd. If not specified, the default configuration is used. See man teamd.conf for the format details.

Format: string

link-watchers

Link watchers configuration for the connection: each link watcher is defined by a dictionary, whose keys depend upon the selected link watcher. Available link watchers are 'ethtool', 'nsna_ping' and 'arp_ping' and it is specified in the dictionary with the key 'name'. Available keys are: ethtool: 'delay-up', 'delay-down', 'init-wait'; nsna_ping: 'init-wait', 'interval', 'missed-max', 'target-host'; arp_ping: all the ones in nsna_ping and 'source-host', 'validate-active', 'validate-inactive', 'send-always'. See teamd.conf man for more details.

Format: array of vardict

mcast-rejoin-count

Corresponds to the teamd mcast_rejoin.count.

Format: int32

mcast-rejoin-interval

Corresponds to the teamd mcast_rejoin.interval.

Format: int32

notify-peers-count

Corresponds to the teamd notify_peers.count.

Format: int32

notify-peers-interval

Corresponds to the teamd notify_peers.interval.

Format: int32

runner

Corresponds to the teamd runner.name. Permitted values are: "roundrobin", "broadcast", "activebackup", "loadbalance", "lacp", "random".

Format: string

runner-active

Corresponds to the teamd runner.active.

Format: boolean

runner-agg-select-policy

Corresponds to the teamd runner.agg_select_policy.

Format: string

runner-fast-rate

Corresponds to the teamd runner.fast_rate.

Format: boolean

runner-hwaddr-policy

Corresponds to the teamd runner.hwaddr_policy.

Format: string

runner-min-ports

Corresponds to the teamd runner.min_ports.

Format: int32

runner-sys-prio

Corresponds to the teamd runner.sys_prio.

Format: int32

runner-tx-balancer

Corresponds to the teamd runner.tx_balancer.name.

Format: string

runner-tx-balancer-interval

Corresponds to the teamd runner.tx_balancer.interval.

Format: int32

runner-tx-hash

Corresponds to the teamd runner.tx_hash.

Format: array of string

team-port setting

Team Port Settings.

Properties:

config

Alias: config

The JSON configuration for the team port. The property should contain raw JSON configuration data suitable for teamd, because the value is passed directly to teamd. If not specified, the default configuration is used. See man teamd.conf for the format details.

Format: string

lacp-key

Corresponds to the teamd ports.PORTIFNAME.lacp_key.

Format: int32

lacp-prio

Corresponds to the teamd ports.PORTIFNAME.lacp_prio.

Format: int32

link-watchers

Link watchers configuration for the connection: each link watcher is defined by a dictionary, whose keys depend upon the selected link watcher. Available link watchers are 'ethtool', 'nsna_ping' and 'arp_ping' and it is specified in the dictionary with the key 'name'. Available keys are: ethtool: 'delay-up', 'delay-down', 'init-wait'; nsna_ping: 'init-wait', 'interval', 'missed-max', 'target-host'; arp_ping: all the ones in nsna_ping and 'source-host', 'validate-active', 'validate-inactive', 'send-always'. See teamd.conf man for more details.

Format: array of vardict

prio

Corresponds to the teamd ports.PORTIFNAME.prio.

Format: int32

queue-id

Corresponds to the teamd ports.PORTIFNAME.queue_id. When set to -1 means the parameter is skipped from the json config.

Format: int32

sticky

Corresponds to the teamd ports.PORTIFNAME.sticky.

Format: boolean

tun setting

Tunnel Settings.

Properties:

group

Alias: group

The group ID which will own the device. If set to NULL everyone will be able to use the device.

Format: string

mode

Alias: mode

The operating mode of the virtual device. Allowed values are NM_SETTING_TUN_MODE_TUN (1) to create a layer 3 device and NM_SETTING_TUN_MODE_TAP (2) to create an Ethernet-like layer 2 one.

Format: uint32

multi-queue

Alias: multi-queue

If the property is set to TRUE, the interface will support multiple file descriptors (queues) to parallelize packet sending or receiving. Otherwise, the interface will only support a single queue.

Format: boolean

owner

Alias: owner

The user ID which will own the device. If set to NULL everyone will be able to use the device.

Format: string

pi

Alias: pi

If TRUE the interface will prepend a 4 byte header describing the physical interface to the packets.

Format: boolean

vnet-hdr

Alias: vnet-hdr

If TRUE the IFF_VNET_HDR the tunnel packets will include a virtio network header.

Format: boolean

vlan setting

VLAN Settings.

Properties:

egress-priority-map

Alias: egress

For outgoing packets, a list of mappings from Linux SKB priorities to 802.1p priorities. The mapping is given in the format "from:to" where both "from" and "to" are unsigned integers, ie "7:3".

Format: array of string

flags

Alias: flags

One or more flags which control the behavior and features of the VLAN interface. Flags include NM_VLAN_FLAG_REORDER_HEADERS (0x1) (reordering of output packet headers), NM_VLAN_FLAG_GVRP (0x2) (use of the GVRP protocol), and NM_VLAN_FLAG_LOOSE_BINDING (0x4) (loose binding of the interface to its master device's operating state). NM_VLAN_FLAG_MVRP (0x8) (use of the MVRP protocol). The default value of this property is NM_VLAN_FLAG_REORDER_HEADERS, but it used to be 0. To preserve backward compatibility, the default-value in the D-Bus API continues to be 0 and a missing property on D-Bus is still considered as 0.

Format: NMVlanFlags (uint32)

id

Alias: id

The VLAN identifier that the interface created by this connection should be assigned. The valid range is from 0 to 4094, without the reserved id 4095.

Format: uint32

ingress-priority-map

Alias: ingress

For incoming packets, a list of mappings from 802.1p priorities to Linux SKB priorities. The mapping is given in the format "from:to" where both "from" and "to" are unsigned integers, ie "7:3".

Format: array of string

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID from which this VLAN interface should be created. If this property is not specified, the connection must contain an "802-3-ethernet" setting with a "mac-address" property.

Format: string

vpn setting

VPN Settings.

Properties:

data

Dictionary of key/value pairs of VPN plugin specific data. Both keys and values must be strings.

Format: dict of string to string

persistent

If the VPN service supports persistence, and this property is TRUE, the VPN will attempt to stay connected across link changes and outages, until explicitly disconnected.

Format: boolean

secrets

Dictionary of key/value pairs of VPN plugin specific secrets like passwords or private keys. Both keys and values must be strings.

Format: dict of string to string

service-type

Alias: vpn-type

D-Bus service name of the VPN plugin that this setting uses to connect to its network. i.e. org.freedesktop.NetworkManager.vpnc for the vpnc plugin.

Format: string

timeout

Timeout for the VPN service to establish the connection. Some services may take quite a long time to connect. Value of 0 means a default timeout, which is 60 seconds (unless overridden by vpn.timeout in configuration file). Values greater than zero mean timeout in seconds.

Format: uint32

user-name

Alias: user

If the VPN connection requires a user name for authentication, that name should be provided here. If the connection is available to more than one user, and the VPN requires each user to supply a different name, then leave this property empty. If this property is empty, NetworkManager will automatically supply the username of the user which requested the VPN connection.

Format: string

vrf setting

VRF settings.

Properties:

table

Alias: table

The routing table for this VRF.

Format: uint32

vxlan setting

VXLAN Settings.

Properties:

ageing

Specifies the lifetime in seconds of FDB entries learnt by the kernel.

Format: uint32

destination-port

Alias: destination-port

Specifies the UDP destination port to communicate to the remote VXLAN tunnel endpoint.

Format: uint32

id

Alias: id

Specifies the VXLAN Network Identifier (or VXLAN Segment Identifier) to use.

Format: uint32

l2-miss

Specifies whether netlink LL ADDR miss notifications are generated.

Format: boolean

l3-miss

Specifies whether netlink IP ADDR miss notifications are generated.

Format: boolean

learning

Specifies whether unknown source link layer addresses and IP addresses are entered into the VXLAN device forwarding database.

Format: boolean

limit

Specifies the maximum number of FDB entries. A value of zero means that the kernel will store unlimited entries.

Format: uint32

local

Alias: local

If given, specifies the source IP address to use in outgoing packets.

Format: string

parent

Alias: dev

If given, specifies the parent interface name or parent connection UUID.

Format: string

proxy

Specifies whether ARP proxy is turned on.

Format: boolean

remote

Alias: remote

Specifies the unicast destination IP address to use in outgoing packets when the destination link layer address is not known in the VXLAN device forwarding database, or the multicast IP address to join.

Format: string

rsc

Specifies whether route short circuit is turned on.

Format: boolean

source-port-max

Alias: source-port-max

Specifies the maximum UDP source port to communicate to the remote VXLAN tunnel endpoint.

Format: uint32

source-port-min

Alias: source-port-min

Specifies the minimum UDP source port to communicate to the remote VXLAN tunnel endpoint.

Format: uint32

tos

Specifies the TOS value to use in outgoing packets.

Format: uint32

ttl

Specifies the time-to-live value to use in outgoing packets.

Format: uint32

wifi-p2p setting

Wi-Fi P2P Settings.

Properties:

peer

Alias: peer

The P2P device that should be connected to. Currently, this is the only way to create or join a group.

Format: string

wfd-ies

The Wi-Fi Display (WFD) Information Elements (IEs) to set. Wi-Fi Display requires a protocol specific information element to be set in certain Wi-Fi frames. These can be specified here for the purpose of establishing a connection. This setting is only useful when implementing a Wi-Fi Display client.

Format: byte array

wps-method

Flags indicating which mode of WPS is to be used. There's little point in changing the default setting as NetworkManager will automatically determine the best method to use.

Format: uint32

wimax setting

WiMax Settings.

Properties:

mac-address

Alias: mac

If specified, this connection will only apply to the WiMAX device whose MAC address matches. This property does not change the MAC address of the device (known as MAC spoofing). Deprecated: 1

Format: byte array

network-name

Alias: nsp

Network Service Provider (NSP) name of the WiMAX network this connection should use.

Deprecated: 1

Format: string

802-3-ethernet setting

Alias: ethernet

Wired Ethernet Settings.

Properties:

accept-all-mac-addresses

When TRUE, setup the interface to accept packets for all MAC addresses. This is enabling the kernel interface flag IFF_PROMISC. When FALSE, the interface will only accept the packets with the interface destination mac address or broadcast.

Format: NMternary (int32)

auto-negotiate

When TRUE, enforce auto-negotiation of speed and duplex mode. If "speed" and "duplex" properties are both specified, only that single mode will be advertised and accepted during the link auto-negotiation process: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabits modes, as in these cases link negotiation is mandatory. When FALSE, "speed" and "duplex" properties should be both set or link configuration will be skipped.

Format: boolean

cloned-mac-address

Alias: cloned-mac

If specified, request that the device use this MAC address instead. This is known as MAC cloning or spoofing. Beside explicitly specifying a MAC address, the special values "preserve", "permanent", "random" and "stable" are supported. "preserve" means not to touch the MAC address on activation. "permanent" means to use the permanent hardware address if the device has one (otherwise this is treated as "preserve"). "random" creates a random MAC address on each connect. "stable" creates a hashed MAC address based on connection.stable-id and a machine dependent key. If unspecified, the value can be overwritten via global defaults, see manual of NetworkManager.conf. If still unspecified, it defaults to "preserve" (older versions of NetworkManager may use a different default value). On D-Bus, this field is expressed as "assigned-mac-address" or the deprecated "cloned-mac-address".

Format: byte array

duplex

When a value is set, either "half" or "full", configures the device to use the specified duplex mode. If "auto-negotiate" is "yes" the specified duplex mode will be the only one advertised during link negotiation: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabits modes, as in these cases link negotiation is mandatory. If the value is unset (the default), the link

configuration will be either skipped (if "auto-negotiate" is "no", the default) or will be auto-negotiated (if "auto-negotiate" is "yes") and the local device will advertise all the supported duplex modes. Must be set together with the "speed" property if specified. Before specifying a duplex mode be sure your device supports it.

Format: string

generate-mac-address-mask

With "cloned-mac-address" setting "random" or "stable", by default all bits of the MAC address are scrambled and a locally-administered, unicast MAC address is created. This property allows to specify that certain bits are fixed. Note that the least significant bit of the first MAC address will always be unset to create a unicast MAC address. If the property is NULL, it is eligible to be overwritten by a default connection setting. If the value is still NULL or an empty string, the default is to create a locally-administered, unicast MAC address. If the value contains one MAC address, this address is used as mask. The set bits of the mask are to be filled with the current MAC address of the device, while the unset bits are subject to randomization. Setting "FE:FF:FF:00:00:00" means to preserve the OUI of the current MAC address and only randomize the lower 3 bytes using the "random" or "stable" algorithm. If the value contains one additional MAC address after the mask, this address is used instead of the current MAC address to fill the bits that shall not be randomized. For example, a value of "FE:FF:FF:00:00:00 68:F7:28:00:00:00" will set the OUI of the MAC address to 68:F7:28, while the lower bits are randomized. A value of "02:00:00:00:00 00:00:00:00:00:00" will create a fully scrambled globally-administered, burned-in MAC address. If the value contains more than one additional MAC addresses, one of them is chosen randomly. For example, "02:00:00:00:00:00 00:00:00:00:00:00 02:00:00:00:00:00" will create a fully scrambled MAC address, randomly locally or globally administered.

Format: string

mac-address

Alias: mac

If specified, this connection will only apply to the Ethernet device whose permanent MAC address matches. This property does not change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mac-address-blacklist

If specified, this connection will never apply to the Ethernet device whose permanent MAC address matches an address in the list. Each MAC address is in the standard hex-digits-and-colons notation (00:11:22:33:44:55).

Format: array of string

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple Ethernet frames.

Format: uint32

port

Specific port type to use if the device supports multiple attachment methods. One of "tp" (Twisted Pair), "aui" (Attachment Unit Interface), "bnc" (Thin Ethernet) or "mii" (Media Independent Interface). If the device supports only one port type, this setting is ignored.

Format: string

s390-nettype

s390 network device type; one of "qeth", "lcs", or "ctc", representing the different types of virtual network devices available on s390 systems.

Format: string

s390-options

Dictionary of key/value pairs of s390-specific device options. Both keys and values must be strings. Allowed keys include "portno", "layer2", "portname", "protocol", among others. Key names must contain only alphanumeric characters (ie, [a-zA-Z0-9]). Currently, NetworkManager itself does nothing with this information. However, s390utils ships a udev rule which parses this information and applies it to the interface.

Format: dict of string to string

s390-subchannels

Identifies specific subchannels that this network device uses for communication with z/VM or s390 host. Like the "mac-address" property for non-z/VM devices, this property can be used to ensure this connection only applies to the network device that uses these subchannels. The list should contain exactly 3 strings, and each string may only be composed of hexadecimal characters and the period (.) character.

Format: array of string

speed

When a value greater than 0 is set, configures the device to use the specified speed. If "auto-negotiate" is "yes" the specified speed will be the only one advertised during link negotiation: this works only for BASE-T 802.3 specifications and is useful for enforcing gigabit speeds, as in this case link negotiation is mandatory. If the value is unset (0, the default), the link configuration will be either skipped (if "auto-negotiate" is "no", the default) or will be auto-negotiated (if "auto-negotiate" is "yes") and the local device will advertise all the supported speeds. In Mbit/s, ie 100 == 100Mbit/s. Must be set together with the "duplex" property when non-zero. Before specifying a speed value be sure your device supports it.

Format: uint32

wake-on-lan

The NMSettingWiredWakeOnLan options to enable. Not all devices support all options. May be any combination of NM_SETTING_WIRED_WAKE_ON_LAN_PHY (0x2), NM_SETTING_WIRED_WAKE_ON_LAN_UNICAST (0x4), NM_SETTING_WIRED_WAKE_ON_LAN_MULTICAST (0x8), NM_SETTING_WIRED_WAKE_ON_LAN_BROADCAST (0x10), NM_SETTING_WIRED_WAKE_ON_LAN_ARP (0x20), NM_SETTING_WIRED_WAKE_ON_LAN_MAGIC (0x40) or the special values NM_SETTING_WIRED_WAKE_ON_LAN_DEFAULT (0x1) (to use global settings) and NM_SETTING_WIRED_WAKE_ON_LAN_IGNORE (0x8000) (to disable management of Wake-on-LAN in NetworkManager).

Format: uint32

wake-on-lan-password

If specified, the password used with magic-packet-based Wake-on-LAN, represented as an Ethernet MAC address. If NULL, no password will be required.

Format: string

wireguard setting

WireGuard Settings.

Properties:

fwmark

The use of fwmark is optional and is by default off. Setting it to 0 disables it. Otherwise, it is a 32-bit fwmark for outgoing packets. Note that "ip4-auto-default-route" or "ip6-auto-default-route" enabled, implies to automatically choose a fwmark.

Format: uint32

ip4-auto-default-route

Whether to enable special handling of the IPv4 default route. If enabled, the IPv4 default route from wireguard.peer-routes will be placed to a dedicated routing-table and two policy routing rules will be added. The fwmark number is also used as routing-table for the default-route, and if fwmark is zero, an unused fwmark/table is chosen automatically. This corresponds to what wg-quick does with Table=auto and what WireGuard calls "Improved Rule-based Routing". Note that for this automatism to work, you usually don't want to set ipv4.gateway, because that will result in a conflicting default route. Leaving this at the default will enable this option automatically if ipv4.never-default is not set and there are any peers that use a default-route as allowed-ips.

Format: NMternary (int32)

ip6-auto-default-route

Like ip4-auto-default-route, but for the IPv6 default route.

Format: NMternary (int32)

listen-port

The listen-port. If listen-port is not specified, the port will be chosen randomly when the interface comes up.

Format: uint32

mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple fragments. If zero a default MTU is used. Note that contrary to wg-quick's MTU setting, this does not take into account the current routes at the time of activation.

Format: uint32

peer-routes

Whether to automatically add routes for the AllowedIPs ranges of the peers. If TRUE (the default), NetworkManager will automatically add routes in the routing tables according to ipv4.route-table and ipv6.route-table. Usually you want this automatism enabled. If FALSE, no such routes are added automatically. In this case, the user may want to configure static routes in ipv4.routes and ipv6.routes, respectively. Note that if the peer's AllowedIPs is "0.0.0.0/0" or "::/0" and the profile's ipv4.never-default or ipv6.never-default setting is enabled, the peer route for this peer won't be added automatically.

Format: boolean

private-key

The 256 bit private-key in base64 encoding.

Format: string

private-key-flags

Flags indicating how to handle the "private-key" property. See the section called "Secret flag types:"

for flag values.

Format: NMSettingSecretFlags (uint32)

802-11-wireless setting

Alias: wifi

Wi-Fi Settings.

Properties:

ap-isolation

Configures AP isolation, which prevents communication between wireless devices connected to this AP. This property can be set to a value different from NM_TERNARY_DEFAULT (-1) only when the interface is configured in AP mode. If set to NM_TERNARY_TRUE (1), devices are not able to communicate with each other. This increases security because it protects devices against attacks from other clients in the network. At the same time, it prevents devices to access resources on the same wireless networks as file shares, printers, etc. If set to NM_TERNARY_FALSE (0), devices can talk to each other. When set to NM_TERNARY_DEFAULT (-1), the global default is used; in case the global default is unspecified it is assumed to be NM_TERNARY_FALSE (0).

Format: NMternary (int32)

band

802.11 frequency band of the network. One of "a" for 5GHz 802.11a or "bg" for 2.4GHz 802.11. This will lock associations to the Wi-Fi network to the specific band, i.e. if "a" is specified, the device will not associate with the same network in the 2.4GHz band even if the network's settings are compatible. This setting depends on specific driver capability and may not work with all drivers.

Format: string

bssid

If specified, directs the device to only associate with the given access point. This capability is highly driver dependent and not supported by all devices. Note: this property does not control the BSSID used when creating an Ad-Hoc network and is unlikely to in the future.

Format: byte array

channel

Wireless channel to use for the Wi-Fi connection. The device will only join (or create for Ad-Hoc networks) a Wi-Fi network on the specified channel. Because channel numbers overlap between bands, this property also requires the "band" property to be set.

Format: uint32

cloned-mac-address

Alias: cloned-mac

If specified, request that the device use this MAC address instead. This is known as MAC cloning or spoofing. Beside explicitly specifying a MAC address, the special values "preserve", "permanent", "random" and "stable" are supported. "preserve" means not to touch the MAC address on activation. "permanent" means to use the permanent hardware address of the device. "random" creates a random MAC address on each connect. "stable" creates a hashed MAC address based on connection.stable-id and a machine dependent key. If unspecified, the value can be overwritten via global defaults, see manual of NetworkManager.conf. If still unspecified, it defaults to "preserve" (older versions of NetworkManager may use a different default value). On D-Bus, this field is expressed as "assigned-mac-address" or the deprecated "cloned-mac-address".

Format: byte array

generate-mac-address-mask

With "cloned-mac-address" setting "random" or "stable", by default all bits of the MAC address are scrambled and a locally-administered, unicast MAC address is created. This property allows to specify that certain bits are fixed. Note that the least significant bit of the first MAC address will always be unset to create a unicast MAC address. If the property is NULL, it is eligible to be overwritten by a default connection setting. If the value is still NULL or an empty string, the default is to create a locally-administered, unicast MAC address. If the value contains one MAC address, this address is used as mask. The set bits of the mask are to be filled with the current MAC address of the device, while the unset bits are subject to randomization. Setting "FE:FF:FF:00:00:00" means to preserve the OUI of the current MAC address and only randomize the lower 3 bytes using the "random" or "stable" algorithm. If the value contains one additional MAC address after the mask, this address is used instead of the current MAC address to fill the bits that shall not be randomized. For example, a value of "FE:FF:FF:00:00:00 68:F7:28:00:00:00" will set the OUI of the MAC address to 68:F7:28, while the lower bits are randomized. A value of "02:00:00:00:00:00 00:00:00:00:00:00" will create a fully scrambled globally-administered, burned-in MAC address. If the value contains more than one additional MAC addresses, one of them is chosen randomly. For example, "02:00:00:00:00:00 00:00:00:00:00:00 02:00:00:00:00:00" will create a fully scrambled MAC address, randomly locally or globally administered.

Format: string

hidden

If TRUE, indicates that the network is a non-broadcasting network that hides its SSID. This works both in infrastructure and AP mode. In infrastructure mode, various workarounds are used for a more reliable discovery of hidden networks, such as probe-scanning the SSID. However, these workarounds expose inherent insecurities with hidden SSID networks, and thus hidden SSID networks should be used with caution. In AP mode, the created network does not broadcast its SSID. Note that marking the network as hidden may be a privacy issue for you (in infrastructure mode) or client stations (in AP mode), as the explicit probe-scans are distinctly recognizable on the air.

Format: boolean

mac-address

Alias: mac

If specified, this connection will only apply to the Wi-Fi device whose permanent MAC address matches. This property does not change the MAC address of the device (i.e. MAC spoofing).

Format: byte array

mac-address-blacklist

A list of permanent MAC addresses of Wi-Fi devices to which this connection should never apply. Each MAC address should be given in the standard hex-digits-and-colons notation (eg "00:11:22:33:44:55").

Format: array of string

mac-address-randomization

One of NM_SETTING_MAC_RANDOMIZATION_DEFAULT (0) (never randomize unless the user has set a global default to randomize and the supplicant supports randomization), NM_SETTING_MAC_RANDOMIZATION_NEVER (1) (never randomize the MAC address), or NM_SETTING_MAC_RANDOMIZATION_ALWAYS (2) (always randomize the MAC address). This property is deprecated for 'cloned-mac-address'. Deprecated: 1

Format: uint32

mode

Alias: mode

Wi-Fi network mode; one of "infrastructure", "mesh", "adhoc" or "ap". If blank, infrastructure is assumed.

Format: string

mtu

Alias: mtu

If non-zero, only transmit packets of the specified size or smaller, breaking larger packets up into multiple Ethernet frames.

Format: uint32

powersave

One of NM_SETTING_WIRELESS_POWERSAVE_DISABLE (2) (disable Wi-Fi power saving), NM_SETTING_WIRELESS_POWERSAVE_ENABLE (3) (enable Wi-Fi power saving), NM_SETTING_WIRELESS_POWERSAVE_IGNORE (1) (don't touch currently configured setting) or NM_SETTING_WIRELESS_POWERSAVE_DEFAULT (0) (use the globally configured value). All other values are reserved.

Format: uint32

rate

If non-zero, directs the device to only use the specified bitrate for communication with the access point. Units are in Kb/s, ie 5500 = 5.5 Mbit/s. This property is highly driver dependent and not all devices support setting a static bitrate.

Format: uint32

seen-bssids

A list of BSSIDs (each BSSID formatted as a MAC address like "00:11:22:33:44:55") that have been detected as part of the Wi-Fi network. NetworkManager internally tracks previously seen BSSIDs. The property is only meant for reading and reflects the BSSID list of NetworkManager. The changes you make to this property will not be preserved.

Format: array of string

ssid

Alias: ssid

SSID of the Wi-Fi network. Must be specified.

Format: byte array

tx-power

If non-zero, directs the device to use the specified transmit power. Units are dBm. This property is highly driver dependent and not all devices support setting a static transmit power.

Format: uint32

wake-on-wlan

The NMSettingWirelessWakeOnWlan options to enable. Not all devices support all options. May be any combination of NM_SETTING_WIRELESS_WAKE_ON_WLAN_ANY (0x2), NM_SETTING_WIRELESS_WAKE_ON_WLAN_DISCONNECT (0x4), NM_SETTING_WIRELESS_WAKE_ON_WLAN_MAGIC (0x8), NM_SETTING_WIRELESS_WAKE_ON_WLAN_GTK_REKEY_FAILURE (0x10), NM_SETTING_WIRELESS_WAKE_ON_WLAN_EAP_IDENTITY_REQUEST (0x20),

NM_SETTING_WIRELESS_WAKE_ON_WLAN_4WAY_HANDSHAKE (0x40),
 NM_SETTING_WIRELESS_WAKE_ON_WLAN_RFKILL_RELEASE (0x80),
 NM_SETTING_WIRELESS_WAKE_ON_WLAN_TCP (0x100) or the special values
 NM_SETTING_WIRELESS_WAKE_ON_WLAN_DEFAULT (0x1) (to use global settings) and
 NM_SETTING_WIRELESS_WAKE_ON_WLAN_IGNORE (0x8000) (to disable management of
 Wake-on-LAN in NetworkManager).

Format: uint32

802-11-wireless-security setting

Alias: wifi-sec

Wi-Fi Security Settings.

Properties:

auth-alg

When WEP is used (ie, key-mgmt = "none" or "ieee8021x") indicate the 802.11 authentication algorithm required by the AP here. One of "open" for Open System, "shared" for Shared Key, or "leap" for Cisco LEAP. When using Cisco LEAP (ie, key-mgmt = "ieee8021x" and auth-alg = "leap") the "leap-username" and "leap-password" properties must be specified.

Format: string

fils

Indicates whether Fast Initial Link Setup (802.11ai) must be enabled for the connection. One of NM_SETTING_WIRELESS_SECURITY_FILS_DEFAULT (0) (use global default value), NM_SETTING_WIRELESS_SECURITY_FILS_DISABLE (1) (disable FILS), NM_SETTING_WIRELESS_SECURITY_FILS_OPTIONAL (2) (enable FILS if the supplicant and the access point support it) or NM_SETTING_WIRELESS_SECURITY_FILS_REQUIRED (3) (enable FILS and fail if not supported). When set to NM_SETTING_WIRELESS_SECURITY_FILS_DEFAULT (0) and no global default is set, FILS will be optionally enabled.

Format: int32

group

A list of group/broadcast encryption algorithms which prevents connections to Wi-Fi networks that do not utilize one of the algorithms in the list. For maximum compatibility leave this property empty. Each list element may be one of "wep40", "wep104", "tkip", or "ccmp".

Format: array of string

key-mgmt

Key management used for the connection. One of "none" (WEP or no password protection), "ieee8021x" (Dynamic WEP), "owe" (Opportunistic Wireless Encryption), "wpa-psk" (WPA2 + WPA3 personal), "sae" (WPA3 personal only), "wpa-eap" (WPA2 + WPA3 enterprise) or "wpa-eap-suite-b-192" (WPA3 enterprise only). This property must be set for any Wi-Fi connection that uses security.

Format: string

leap-password

The login password for legacy LEAP connections (ie, key-mgmt = "ieee8021x" and auth-alg = "leap").

Format: string

leap-password-flags

Flags indicating how to handle the "leap-password" property. See the section called "Secret flag

types:” for flag values.

Format: NMSettingSecretFlags (uint32)

leap-username

The login username for legacy LEAP connections (ie, key-mgmt = "ieee8021x" and auth-alg = "leap").

Format: string

pairwise

A list of pairwise encryption algorithms which prevents connections to Wi-Fi networks that do not utilize one of the algorithms in the list. For maximum compatibility leave this property empty. Each list element may be one of "tkip" or "ccmp".

Format: array of string

pmf

Indicates whether Protected Management Frames (802.11w) must be enabled for the connection. One of NM_SETTING_WIRELESS_SECURITY_PMF_DEFAULT (0) (use global default value), NM_SETTING_WIRELESS_SECURITY_PMF_DISABLE (1) (disable PMF), NM_SETTING_WIRELESS_SECURITY_PMF_OPTIONAL (2) (enable PMF if the supplicant and the access point support it) or NM_SETTING_WIRELESS_SECURITY_PMF_REQUIRED (3) (enable PMF and fail if not supported). When set to NM_SETTING_WIRELESS_SECURITY_PMF_DEFAULT (0) and no global default is set, PMF will be optionally enabled.

Format: int32

proto

List of strings specifying the allowed WPA protocol versions to use. Each element may be one "wpa" (allow WPA) or "rsn" (allow WPA2/RSN). If not specified, both WPA and RSN connections are allowed.

Format: array of string

psk

Pre-Shared-Key for WPA networks. For WPA-PSK, it's either an ASCII passphrase of 8 to 63 characters that is (as specified in the 802.11i standard) hashed to derive the actual key, or the key in form of 64 hexadecimal character. The WPA3-Personal networks use a passphrase of any length for SAE authentication.

Format: string

psk-flags

Flags indicating how to handle the "psk" property. See the section called “Secret flag types:” for flag values.

Format: NMSettingSecretFlags (uint32)

wep-key-flags

Flags indicating how to handle the "wep-key0", "wep-key1", "wep-key2", and "wep-key3" properties. See the section called “Secret flag types:” for flag values.

Format: NMSettingSecretFlags (uint32)

wep-key-type

Controls the interpretation of WEP keys. Allowed values are NM_WEP_KEY_TYPE_KEY (1), in which case the key is either a 10- or 26-character hexadecimal string, or a 5- or 13-character ASCII

password; or NM_WEP_KEY_TYPE_PASSPHRASE (2), in which case the passphrase is provided as a string and will be hashed using the de-facto MD5 method to derive the actual WEP key.

Format: NMWepKeyType (uint32)

wep-key0

Index 0 WEP key. This is the WEP key used in most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key1

Index 1 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key2

Index 2 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-key3

Index 3 WEP key. This WEP index is not used by most networks. See the "wep-key-type" property for a description of how this key is interpreted.

Format: string

wep-tx-keyidx

When static WEP is used (ie, key-mgmt = "none") and a non-default WEP key index is used by the AP, put that WEP key index here. Valid values are 0 (default key) through 3. Note that some consumer access points (like the Linksys WRT54G) number the keys 1 – 4.

Format: uint32

wps-method

Flags indicating which mode of WPS is to be used if any. There's little point in changing the default setting as NetworkManager will automatically determine whether it's feasible to start WPS enrollment from the Access Point capabilities. WPS can be disabled by setting this property to a value of 1.

Format: uint32

wpan setting

IEEE 802.15.4 (WPAN) MAC Settings.

Properties:

channel

Alias: channel

IEEE 802.15.4 channel. A positive integer or -1, meaning "do not set, use whatever the device is already set to".

Format: int32

mac-address

Alias: mac

If specified, this connection will only apply to the IEEE 802.15.4 (WPAN) MAC layer device whose

permanent MAC address matches.

Format: string

page

Alias: page

IEEE 802.15.4 channel page. A positive integer or -1, meaning "do not set, use whatever the device is already set to".

Format: int32

pan-id

Alias: pan-id

IEEE 802.15.4 Personal Area Network (PAN) identifier.

Format: uint32

short-address

Alias: short-addr

Short IEEE 802.15.4 address to be used within a restricted environment.

Format: uint32

bond-port setting

Bond Port Settings.

Properties:

queue-id

Alias: queue-id

The queue ID of this bond port. The maximum value of queue ID is the number of TX queues currently active in device.

Format: uint32

hostname setting

Hostname settings.

Properties:

from-dhcp

Whether the system hostname can be determined from DHCP on this connection. When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_TRUE (1).

Format: NMternary (int32)

from-dns-lookup

Whether the system hostname can be determined from reverse DNS lookup of addresses on this device. When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_TRUE (1).

Format: NMternary (int32)

only-from-default

If set to NM_TERNARY_TRUE (1), NetworkManager attempts to get the hostname via DHCPv4/DHCPv6 or reverse DNS lookup on this device only when the device has the default route for the given address family (IPv4/IPv6). If set to NM_TERNARY_FALSE (0), the hostname can be set from this device even if it doesn't have the default route. When set to NM_TERNARY_DEFAULT (-1), the value from global configuration is used. If the property doesn't have a value in the global configuration, NetworkManager assumes the value to be NM_TERNARY_FALSE (0).

Format: NMternary (int32)

priority

The relative priority of this connection to determine the system hostname. A lower numerical value is better (higher priority). A connection with higher priority is considered before connections with lower priority. If the value is zero, it can be overridden by a global value from NetworkManager configuration. If the property doesn't have a value in the global configuration, the value is assumed to be 100. Negative values have the special effect of excluding other connections with a greater numerical priority value; so in presence of at least one negative priority, only connections with the lowest priority value will be used to determine the hostname.

Format: int32

veth setting

Veth Settings.

Properties:

peer

Alias: peer

This property specifies the peer interface name of the veth. This property is mandatory.

Format: string

Secret flag types:

Each password or secret property in a setting has an associated *flags* property that describes how to handle that secret. The *flags* property is a bitfield that contains zero or more of the following values logically OR-ed together.

- 0x0 (none) – the system is responsible for providing and storing this secret. This may be required so that secrets are already available before the user logs in. It also commonly means that the secret will be stored in plain text on disk, accessible to root only. For example via the keyfile settings plugin as described in the "PLUGINS" section in **NetworkManager.conf(5)**.
- 0x1 (agent-owned) – a user-session secret agent is responsible for providing and storing this secret; when it is required, agents will be asked to provide it.
- 0x2 (not-saved) – this secret should not be saved but should be requested from the user each time it is required. This flag should be used for One-Time-Pad secrets, PIN codes from hardware tokens, or if the user simply does not want to save the secret.
- 0x4 (not-required) – in some situations it cannot be automatically determined that a secret is required or not. This flag hints that the secret is not required and should not be requested from the user.

FILES

/etc/NetworkManager/system-connections or distro plugin-specific location

SEE ALSO

nmcli(1), **nmcli-examples(7)**, **NetworkManager(8)**, **nm-settings-dbus(5)**, **nm-settings-keyfile(5)**, **NetworkManager.conf(5)**

NAME

nmcli-examples – usage examples of nmcli

SYNOPSIS

nmcli [OPTIONS...]

DESCRIPTION

nmcli is a command-line client for NetworkManager. It allows controlling NetworkManager and reporting its status. For more information please refer to **nmcli(1)** manual page.

The purpose of this manual page is to provide you with various examples and usage scenarios of *nmcli*.

EXAMPLES**Example 1. Listing available Wi-Fi APs**

```
$ nmcli device wifi list
* SSID      MODE  CHAN RATE   SIGNAL BARS SECURITY
  netdatacomm_local Infra 6 54 Mbit/s 37   __ WEP
* F1        Infra 11 54 Mbit/s 98   WPA1
  LoremCorp     Infra 1 54 Mbit/s 62   _ WPA2 802.1X
  Internet     Infra 6 54 Mbit/s 29   __ WPA1
  HPB110a.F2672A Ad-Hoc 6 54 Mbit/s 22   __ --
  Jozinet     Infra 1 54 Mbit/s 19   __ WEP
  VOIP        Infra 1 54 Mbit/s 20   __ WEP
  MARTINA     Infra 4 54 Mbit/s 32   __ WPA2
  N24PU1      Infra 7 11 Mbit/s 22   __ --
  alfa        Infra 1 54 Mbit/s 67   _ WPA2
  bertnet     Infra 5 54 Mbit/s 20   __ WPA1 WPA2
```

This command shows how to list available Wi-Fi networks (APs). You can also use *--fields* option for displaying different columns. **nmcli -f all dev wifi list** will show all of them.

Example 2. Connect to a password-protected wifi network

```
$ nmcli device wifi connect "$SSID" password "$PASSWORD"
```

```
$ nmcli --ask device wifi connect "$SSID"
```

Example 3. Showing general information and properties for a Wi-Fi interface

```
$ nmcli -p -f general,wifi-properties device show wlan0
```

```
=====
Device details (wlan0)
=====

GENERAL.DEVICE:      wlan0
GENERAL.TYPE:        wifi
GENERAL.VENDOR:      Intel Corporation
GENERAL.PRODUCT:     PRO/Wireless 5100 AGN [Shiloh] Network Connection
GENERAL.DRIVER:       iwlwifi
GENERAL.DRIVER-VERSION: 3.8.13-100.fc17.x86_64
GENERAL.FIRMWARE-VERSION: 8.83.5.1 build 33692
GENERAL.HWADDR:      00:1E:65:37:A1:D3
GENERAL.MTU:         1500
GENERAL.STATE:       100 (connected)
GENERAL.REASON:      0 (No reason given)
GENERAL.UDI:         /sys/devices/pci0000:00/0000:00:1c.1/net/wlan0
GENERAL.IP-IFACE:    wlan0
GENERAL.IS-SOFTWARE: no
GENERAL.NM-MANAGED:  yes
```

```

GENERAL.AUTOCONNECT: yes
GENERAL.FIRMWARE-MISSING: no
GENERAL.CONNECTION: My Alfa WiFi
GENERAL.CON-UUID: 85194f4c-d496-4eec-bae0-d880b4cbcf26
GENERAL.CON-PATH: /org/freedesktop/NetworkManager/ActiveConnection/
10

WIFI-PROPERTIES.WEP: yes
WIFI-PROPERTIES.WPA: yes
WIFI-PROPERTIES.WPA2: yes
WIFI-PROPERTIES.TKIP: yes
WIFI-PROPERTIES.CCMP: yes
WIFI-PROPERTIES.AP: no
WIFI-PROPERTIES.ADHOC: yes

```

This command shows information about a Wi-Fi device.

Example 4. Listing NetworkManager polkit permissions

\$ nmcli general permissions

PERMISSION	VALUE
org.freedesktop.NetworkManager.enable-disable-network	yes
org.freedesktop.NetworkManager.enable-disable-wifi	yes
org.freedesktop.NetworkManager.enable-disable-wwan	yes
org.freedesktop.NetworkManager.enable-disable-wimax	yes
org.freedesktop.NetworkManager.sleep-wake	no
org.freedesktop.NetworkManager.network-control	yes
org.freedesktop.NetworkManager.wifi.share.protected	yes
org.freedesktop.NetworkManager.wifi.share.open	yes
org.freedesktop.NetworkManager.settings.modify.system	yes
org.freedesktop.NetworkManager.settings.modify.own	yes
org.freedesktop.NetworkManager.settings.modify.hostname	auth
org.freedesktop.NetworkManager.settings.modify.global-dns	auth
org.freedesktop.NetworkManager.reload	auth

This command shows configured polkit permissions for various NetworkManager operations. These permissions or actions (using polkit language) are configured by a system administrator and are not meant to be changed by users. The usual place for the polkit configuration is `/usr/share/polkit-1/actions/org.freedesktop.NetworkManager.policy`. `pkaction` command can display description for polkit actions.

pkaction --action-id org.freedesktop.NetworkManager.network-control --verbose

More information about polkit can be found at <http://www.freedesktop.org/wiki/Software/polkit>.

Example 5. Listing NetworkManager log level and domains

\$ nmcli general logging

LEVEL DOMAINS

INFO PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,
WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS_PROPS,TEAM,CONCHECK,DC
B,DISPATCH

This command shows current NetworkManager logging status.

Example 6. Changing NetworkManager logging

```
$ nmcli g log level DEBUG domains CORE,ETHER,IP
$ nmcli g log level INFO domains DEFAULT
```

The first command makes NetworkManager log in DEBUG level, and only for CORE, ETHER and IP domains. The second command restores the default logging state. Please refer to the **NetworkManager.conf(5)** manual page for available logging levels and domains.

Example 7. Activating a VPN connection profile requiring interactive password input

```
$ nmcli --ask con up my-vpn-con
```

This command activates a VPN connection profile enabling nmcli to interact with the user ('--ask'): this will allow nmcli to prompt for the VPN password on the command line when the *password-flags* are set to '0x02' ('always ask', see **nm-settings(5)**). This is particularly useful for OTP based VPNs, as the user needs to be prompted for the password each time the connection is activated.

Example 8. Adding a bonding master and two slave connection profiles

```
$ nmcli con add type bond ifname mybond0 mode active-backup
$ nmcli con add type ethernet ifname eth1 master mybond0
$ nmcli con add type ethernet ifname eth2 master mybond0
```

This example demonstrates adding a bond master connection and two slaves. The first command adds a master bond connection, naming the bonding interface *mybond0* and using *active-backup* mode. The next two commands add slaves connections, both enslaved to *mybond0*. The first slave will be bound to *eth1* interface, the second to *eth2*.

Example 9. Adding a team master and two slave connection profiles

```
$ nmcli con add type team con-name Team1 ifname Team1 config team1-master-json.conf
$ nmcli con add type ethernet con-name Team1-slave1 ifname em1 master Team1
$ nmcli con add type ethernet con-name Team1-slave2 ifname em2 master Team1
```

This example demonstrates adding a team master connection profile and two slaves. It is very similar to the bonding example. The first command adds a master team profile, naming the team interface and the profile *Team1*. The team configuration for the master is read from *team1-master-json.conf* file. Later, you can change the configuration with *modify* command (**nmcli con modify Team1 team.config team1-master-another-json.conf**). The last two commands add slaves profiles, both enslaved to *Team1*. The first slave will be bound to the *em1* interface, the second to *em2*. The slaves don't specify *config* and thus *teamd* will use its default configuration. You will activate the whole setup by activating both slaves:

```
$ nmcli con up Team1-slave1
$ nmcli con up Team1-slave2
```

By default, the created profiles are marked for auto-activation. But if another connection has been activated on the device, the new profile won't activate automatically and you need to activate it manually.

Example 10. Adding a bridge and two slave profiles

```
$ nmcli con add type bridge con-name TowerBridge ifname TowerBridge
$ nmcli con add type ethernet con-name br-slave-1 ifname ens3 master TowerBridge
$ nmcli con add type ethernet con-name br-slave-2 ifname ens4 master TowerBridge
$ nmcli con modify TowerBridge bridge.stp no
```

This example demonstrates adding a bridge master connection and two slaves. The first command adds a master bridge connection, naming the bridge interface and the profile as *TowerBridge*. The next two commands add slaves profiles, both will be enslaved to *TowerBridge*. The first slave will be tied to *ens3* interface, the second to *ens4*. The last command will disable 802.1D STP for the *TowerBridge* profile.

Example 11. Adding an ethernet connection profile with manual IP configuration

```
$ nmcli con add con-name my-con-em1 ifname em1 type ethernet \
  ip4 192.168.100.100/24 gw4 192.168.100.1 ip4 1.2.3.4 ip6 abbe::cafe
$ nmcli con mod my-con-em1 ipv4.dns "8.8.8.8 8.8.4.4"
$ nmcli con mod my-con-em1 +ipv4.dns 1.2.3.4
$ nmcli con mod my-con-em1 ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844"
$ nmcli -p con show my-con-em1
```

The first command adds an Ethernet connection profile named *my-con-em1* that is bound to interface name *em1*. The profile is configured with static IP addresses. Three addresses are added, two IPv4 addresses and one IPv6. The first IP 192.168.100.100 has a prefix of 24 (netmask equivalent of 255.255.255.0). Gateway entry will become the default route if this profile is activated on em1 interface (and there is no connection with higher priority). The next two addresses do not specify a prefix, so a default prefix will be used, i.e. 32 for IPv4 and 128 for IPv6. The second, third and fourth commands modify DNS parameters of the new connection profile. The last *con show* command displays the profile so that all parameters can be reviewed.

Example 12. Convenient field values retrieval for scripting

```
$ nmcli -g ip4.address connection show my-con-eth0  
192.168.1.12/24  
  
$ nmcli -g ip4.address,ip4.dns connection show my-con-eth0  
192.168.1.12/24  
192.168.1.1  
  
$ nmcli -g ip4 connection show my-con-eth0  
IP4[192.168.1.12/24,192.168.1.1,192.168.1.1]
```

This example shows retrieval of ip4 connection field values via the `--get-values` option. Multiple comma separated fields can be provided: they will be printed one per line. If a whole section is provided instead of a single field, the name of the section will be printed followed by all the related field values on the same line. See also `--terse`, `--mode`, `--fields` and `--escape` options in **nmcli(1)** manual page for more customized output.

Example 13. Adding an Ethernet connection and configuring SR-IOV VFs

```
$ nmcli con add type ethernet con-name EthernetPF ifname em1
$ nmcli con modify EthernetPF sriov.total-vfs 3 sriov.autoprobe-drivers false
$ nmcli con modify EthernetPF sriov.vfs '0 mac=00:11:22:33:44:55 vlans=10, 1 trust=true spoof-check=false'
$ nmcli con modify EthernetPF +sriov.vfs '2 max-tx-rate=20'
```

This example demonstrates adding an Ethernet connection for physical function (PF) `ens4` and configuring 3 SR-IOV virtual functions (VFs) on it. The first VF is configured with MAC address 00:11:22:33:44:55 and VLAN 10, the second one has the *trust* and *spoof-check* features respectively enabled and disabled. VF number 2 has a maximum transmission rate of 20Mbps. The kernel is instructed to not automatically instantiate a network interface for the VFs.

Example 14. Escaping colon characters in tabular mode

```
$ nmcli -t -f general -e yes -m tab dev show eth0
GENERAL:eth0:ethernet:Intel Corporation:82567LM Gigabit Network Connection:
e1000e:2.1.4-k:1.8-3:00\;22\;68\;15\;29\;21:1500:100 (connected):0 (No reas
on given):\sys\devices\pci0000\00\0000\00\19.0\net\eth0:eth0:yes:yes:no:
ethernet-13:89cbcfc6-dc85-456c-9c8b-bd828fee3917:/org/freedesktop/NetworkMa
nager/ActiveConnection/9
```

This example shows escaping colon characters in tabular mode. It may be useful for script processing,

because ':' is used as a field separator.

Example 15. nmcli usage in a NetworkManager dispatcher script to make Ethernet and Wi-Fi mutually exclusive

```
#!/bin/bash
export LC_ALL=C

enable_disable_wifi () {
    result=$(nmcli dev | grep "ethernet" | grep -w "connected")
    if [ -n "$result" ]; then
        nmcli radio wifi off
    else
        nmcli radio wifi on
    fi
}

if [ "$2" = "up" ]; then
    enable_disable_wifi
fi

if [ "$2" = "down" ]; then
    enable_disable_wifi
fi
```

This dispatcher script makes Wi-Fi mutually exclusive with wired networking. When a wired interface is connected, Wi-Fi will be set to airplane mode (rfkill). When the wired interface is disconnected, Wi-Fi will be turned back on. Name this script e.g. 70-wifi-wired-exclusive.sh and put it into /etc/NetworkManager/dispatcher.d/ directory. See **NetworkManager(8)** manual page for more information about NetworkManager dispatcher scripts.

Example sessions of interactive connection editor

Example 16. Adding an ethernet connection profile in interactive editor (a)

```
$ nmcli connection edit type ethernet
```

```
====| nmcli interactive connection editor |====
```

Adding a new '802-3-ethernet' connection

Type 'help' or '?' for available commands.

Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-1x, ipv4, ipv6, dcb

nmcli> **print**

```
=====
Connection details
=====
connection.id:          ethernet-4
connection.uuid:        de89cdeb-a3e1-4d53-8fa0-c22546c775f4
connection.interface-name:  --
connection.type:         802-3-ethernet
connection.autoconnect:   yes
```

```

connection.autoconnect-priority: 0
connection.timestamp:          0
connection.read-only:          no
connection.permissions:
connection.zone:                --
connection.master:              --
connection.slave-type:          --
connection.secondaries:
connection.gateway-ping-timeout: 0
-----
802-3-ethernet.port:           --
802-3-ethernet.speed:          0
802-3-ethernet.duplex:         --
802-3-ethernet.auto-negotiate: yes
802-3-ethernet.mac-address:    --
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:            auto
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:    --
802-3-ethernet.s390-options:
-----
ipv4.method:                  auto
ipv4.dns:
ipv4.dns-search:
ipv4.addresses:
ipv4.gateway:                 --
ipv4.routes:
ipv4.route-metric:             -1
ipv4.ignore-auto-routes:       no
ipv4.ignore-auto-dns:          no
ipv4.dhcp-client-id:          --
ipv4.dhcp-send-hostname:       yes
ipv4.dhcp-hostname:           --
ipv4.never-default:            no
ipv4.may-fail:                 yes
-----
ipv6.method:                  auto
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:
ipv6.gateway:                 --
ipv6.routes:
ipv6.route-metric:             -1
ipv6.ignore-auto-routes:       no
ipv6.ignore-auto-dns:          no
ipv6.never-default:            no
ipv6.may-fail:                 yes
ipv6.ip6-privacy:              -1 (unknown)
ipv6.dhcp-hostname:           --
-----
nmcli> goto ethernet
You may edit the following properties: port, speed, duplex, auto-negotiate,
mac-address, cloned-mac-address, mac-address-blacklist, mtu, s390-subchann

```

```
els, s390-nettype, s390-options
nmcli 802-3-ethernet> set mtu 1492
nmcli 802-3-ethernet> b
nmcli> goto ipv4.addresses
nmcli ipv4.addresses> desc
```

```
==== [addresses] ====
[NM property description]
Array of IP addresses.
```

[nmcli specific description]
Enter a list of IPv4 addresses formatted as:
ip[/prefix], ip[/prefix],...
Missing prefix is regarded as prefix of 32.

Example: 192.168.1.5/24, 10.0.0.11/24

```
nmcli ipv4.addresses> set 192.168.1.100/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
nmcli ipv4.addresses>
nmcli ipv4.addresses> print
addresses: 192.168.1.100/24
nmcli ipv4.addresses> back
nmcli ipv4> b
nmcli> set ipv4.gateway 192.168.1.1
nmcli> verify
Verify connection: OK
nmcli> print
```

Connection details

```
connection.id:          ethernet-4
connection.uuid:        de89cdeb-a3e1-4d53-8fa0-c22546c775f4
connection.interface-name:  --
connection.type:        802-3-ethernet
connection.autoconnect: yes
connection.autoconnect-priority: 0
connection.timestamp:    0
connection.read-only:    no
connection.permissions:
connection.zone:         --
connection.master:        --
connection.slave-type:    --
connection.secondaries:
connection.gateway-ping-timeout: 0
-----
802-3-ethernet.port:    --
802-3-ethernet.speed:   0
802-3-ethernet.duplex:  --
802-3-ethernet.auto-negotiate: yes
802-3-ethernet.mac-address:  --
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:      1492
```

```

802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:    --
802-3-ethernet.s390-options:

-----
ipv4.method:           manual
ipv4.dns:
ipv4.dns-search:
ipv4.addresses:       192.168.1.100/24
ipv4.gateway:         192.168.1.1
ipv4.routes:
ipv4.route-metric:   -1
ipv4.ignore-auto-routes: no
ipv4.ignore-auto-dns: no
ipv4.dhcp-client-id: --
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname:   --
ipv4.never-default:  no
ipv4.may-fail:        yes

-----
ipv6.method:           auto
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:
ipv6.routes:
ipv6.route-metric:   -1
ipv6.ignore-auto-routes: no
ipv6.ignore-auto-dns: no
ipv6.never-default:  no
ipv6.may-fail:        yes
ipv6.ip6-privacy:    -1 (unknown)
ipv6.dhcp-hostname:  --

-----
nmcli> set ipv4.dns 8.8.8.8 8.8.4.4
nmcli> print
=====
          Connection details
=====
connection.id:           ethernet-4
connection.uuid:         de89cdeb-a3e1-4d53-8fa0-c22546c775f4
connection.interface-name: --
connection.type:          802-3-ethernet
connection.autoconnect:   yes
connection.autoconnect-priority: 0
connection.timestamp:    0
connection.read-only:    no
connection.permissions:
connection.zone:         --
connection.master:        --
connection.slave-type:   --
connection.secondaries:
connection.gateway-ping-timeout: 0

-----
802-3-ethernet.port:    --
802-3-ethernet.speed:   0

```

```

802-3-ethernet.duplex:      --
802-3-ethernet.auto-negotiate: yes
802-3-ethernet.mac-address:   --
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:          1492
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:   --
802-3-ethernet.s390-options:

-----
ipv4.method:                 manual
ipv4.dns:                    8.8.8.8,8.8.4.4
ipv4.dns-search:
ipv4.addresses:              192.168.1.100/24
ipv4.gateway:                192.168.1.1
ipv4.routes:
ipv4.route-metric:           -1
ipv4.ignore-auto-routes:     no
ipv4.ignore-auto-dns:        no
ipv4.dhcp-client-id:         --
ipv4.dhcp-send-hostname:     yes
ipv4.dhcp-hostname:          --
ipv4.never-default:          no
ipv4.may-fail:               yes

-----
ipv6.method:                 auto
ipv6.dns:
ipv6.dns-search:
ipv6.addresses:
ipv6.gateway:                --
ipv6.routes:
ipv6.route-metric:           -1
ipv6.ignore-auto-routes:     no
ipv6.ignore-auto-dns:        no
ipv6.never-default:          no
ipv6.may-fail:               yes
ipv6.ip6-privacy:            -1 (unknown)
ipv6.dhcp-hostname:          --

```

```

nmcli> verify
Verify connection: OK
nmcli> save
Connection 'ethernet-4' (de89cdeb-a3e1-4d53-8fa0-c22546c775f4) successfully
saved.
nmcli> quit

```

Example session in the nmcli interactive connection editor. The scenario creates an Ethernet connection profile with static addressing (IPs and DNS).

Example 17. Bluetooth connection profiles

NetworkManger supports both connecting to NAP and DUN devices as a client. It also supports sharing the network via a NAP server.

For NAP client connections, NetworkManager automatically creates a suitable in-memory profile for paired devices if none is available. You may use that generated profile directly, but you may also modify and persist it, which will prevent it from automatically re-creating it. You may also create a profile from scratch.

For example, the following uses DHCP and IPv6 autoconf for address configuration:

```
$ nmcli connection add type bluetooth con-name "Profile for My Bluetooth Device (NAP)" autoconnect no bluetoo
```

For DUN connections, the user needs to configure modem settings and hence no profile gets created automatically. The modem settings depend on your device and you either need a "gsm" or a "csma" section. For example,

```
$ nmcli connection add type bluetooth con-name "Profile for My Bluetooth Device (DUN)" autoconnect no bluetoo
```

Finally, you can create a bluetooth hotspot. BlueZ implements those as a bridge device, so such profiles also have a bridge section. Also, you probably want to set IP methods as "shared", so that clients get automatic IP addressing. Note that the "shared" IPv4 method requires dnsmasq to be available.

```
$ nmcli connection add type bluetooth con-name "My Bluetooth Hotspot" autoconnect no ifname btnap0 bluetooth
```

SEE ALSO

nmcli(1), NetworkManager(8), NetworkManager.conf(5), nm-settings(5), nm-online(1), nm-applet(1), nm-connection-editor(1)

NAME

nmcli – command-line tool for controlling NetworkManager

SYNOPSIS

```
nmcli [OPTIONS...] {help | general | networking | radio | connection | device | agent | monitor}  
[COMMAND] [ARGUMENTS...]
```

DESCRIPTION

nmcli is a command-line tool for controlling NetworkManager and reporting network status. It can be utilized as a replacement for **nm-applet** or other graphical clients. **nmcli** is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status. See **nmcli-examples(7)** for ready to run nmcli examples.

Typical uses include:

- Scripts: Utilize NetworkManager via **nmcli** instead of managing network connections manually. **nmcli** supports a terse output format which is better suited for script processing. Note that NetworkManager can also execute scripts, called "dispatcher scripts", in response to network events. See **NetworkManager(8)** for details about these dispatcher scripts.
- Servers, headless machines, and terminals: **nmcli** can be used to control NetworkManager without a GUI, including creating, editing, starting and stopping network connections and viewing network status.

OPTIONS

-a | **--ask**

When using this option **nmcli** will stop and ask for any missing required arguments, so do not use this option for non-interactive purposes like scripts. This option controls, for example, whether you will be prompted for a password if it is required for connecting to a network.

-c | **--colors** {yes | no | auto}

This option controls color output (using terminal escape sequences). yes enables colors, no disables them, auto only produces colors when standard output is directed to a terminal. The default value is auto.

The actual colors used are configured as described in **terminal-colors.d(5)**. Please refer to the COLORS section for a list of color names supported by **nmcli**.

If the environment variable NO_COLOR is set (to any value), then coloring is disabled with mode "auto". Explicitly enabling coloring overrides the environment variable.

--complete-args

Instead of conducting the desired action, **nmcli** will list possible completions for the last argument. This is useful to implement argument completion in shell.

The exit status will indicate success or return a code 65 to indicate the last argument is a file name.

NetworkManager ships with command completion support for GNU Bash.

-e | **--escape** {yes | no}

Whether to escape : and \ characters in terse tabular mode. The escape character is \.

If omitted, default is yes.

-f | **--fields** {field1,field2... | all | common}

This option is used to specify what fields (column names) should be printed. Valid field names differ for specific commands. List available fields by providing an invalid value to the **--fields** option. all is used to print all valid field values of the command. common is used to print common field values of the command.

If omitted, default is common.

-g | --get-values {field1,field2... | all | common}

This option is used to print values from specific fields. It is basically a shortcut for --mode tabular --terse --fields and is a convenient way to retrieve values for particular fields. The values are printed one per line without headers.

If a section is specified instead of a field, the section name will be printed followed by colon separated values of the fields belonging to that section, all on the same line.

-h | --help

Print help information.

-m | --mode {tabular | multiline}

Switch between tabular and multiline output:

tabular

Output is a table where each line describes a single entry. Columns define particular properties of the entry.

multiline

Each entry comprises multiple lines, each property on its own line. The values are prefixed with the property name.

If omitted, default is tabular for most commands. For the commands producing more structured information, that cannot be displayed on a single line, default is multiline. Currently, they are:

- nmcli connection show *ID*
- nmcli device show

-p | --pretty

Output is pretty. This causes **nmcli** to produce easily readable outputs for humans, i.e. values are aligned, headers are printed, etc.

-s | --show-secrets

When using this option **nmcli** will display passwords and secrets that might be present in an output of an operation. This option also influences echoing passwords typed by user as an input.

-t | --terse

Output is terse. This mode is designed and suitable for computer (script) processing.

-v | --version

Show **nmcli** version.

-w | --wait seconds

This option sets a timeout period for which **nmcli** will wait for NetworkManager to finish operations. It is especially useful for commands that may take a longer time to complete, e.g. connection activation.

Specifying a value of 0 instructs **nmcli** not to wait but to exit immediately with a status of success. The default value depends on the executed command.

GENERAL COMMANDS

nmcli general {status | hostname | permissions | logging | reload} [ARGUMENTS...]

Use this command to show NetworkManager status and permissions. You can also get and change system hostname, as well as NetworkManager logging level and domains.

status

Show overall status of NetworkManager. This is the default action, when no additional command is provided for **nmcli general**.

hostname [hostname]

Get and change system hostname. With no arguments, this prints currently configured hostname. When you pass a hostname, it will be handed over to NetworkManager to be set as a new system hostname.

Note that the term "system" hostname may also be referred to as "persistent" or "static" by other programs or tools. The hostname is stored in /etc/hostname file in most distributions. For example, systemd-hostnamed service uses the term "static" hostname and it only reads the /etc/hostname file when it starts.

permissions

Show the permissions a caller has for various authenticated operations that NetworkManager provides, like enable and disable networking, changing Wi-Fi and WWAN state, modifying connections, etc.

logging [level level] [domains domains...]

Get and change NetworkManager logging level and domains. Without any argument current logging level and domains are shown. In order to change logging state, provide **level** and, or, **domain** parameters. See **NetworkManager.conf(5)** for available level and domain values.

reload [flags...]

Reload NetworkManager's configuration and perform certain updates, like flushing caches or rewriting external state to disk. This is similar to sending SIGHUP to NetworkManager but it allows for more fine-grained control over what to reload through the flags argument. It also allows non-root access via PolicyKit and contrary to signals it is synchronous. Available flags are:

conf

Reload the NetworkManager.conf configuration from disk. Note that this does not include connections, which can be reloaded through **nmcli connection reload** instead.

dns-rc

Update DNS configuration, which usually involves writing /etc/resolv.conf anew. This is equivalent to sending the SIGUSR1 signal to the NetworkManager process.

dns-full

Restart the DNS plugin. This is for example useful when using dnsmasq plugin, which uses additional configuration in /etc/NetworkManager/dnsmasq.d. If you edit those files, you can restart the DNS plugin. This action shortly interrupts name resolution.

With no flags, everything that is supported is reloaded, which is identical to sending a SIGHUP. See **NetworkManager(8)** for more details about signals.

NETWORKING CONTROL COMMANDS

nmcli networking {on | off | connectivity} [ARGUMENTS...]

Query NetworkManager networking status, enable and disable networking.

on, off

Enable or disable networking control by NetworkManager. All interfaces managed by NetworkManager are deactivated when networking is disabled.

connectivity [check]

Get network connectivity state. The optional **check** argument tells NetworkManager to re-check the connectivity, else the most recent known connectivity state is displayed without re-checking.

Possible states are:

none

the host is not connected to any network.

portal

the host is behind a captive portal and cannot reach the full Internet.

limited

the host is connected to a network, but it has no access to the Internet.

full

the host is connected to a network and has full access to the Internet.

unknown

the connectivity status cannot be found out.

RADIO TRANSMISSION CONTROL COMMANDS

nmcli radio {all | wifi | wwan} [ARGUMENTS...]

Show radio switches status, or enable and disable the switches.

wifi [on | off]

Show or set status of Wi-Fi in NetworkManager. If no arguments are supplied, Wi-Fi status is printed; **on** enables Wi-Fi; **off** disables Wi-Fi.

wwan [on | off]

Show or set status of WWAN (mobile broadband) in NetworkManager. If no arguments are supplied, mobile broadband status is printed; **on** enables mobile broadband, **off** disables it.

all [on | off]

Show or set all previously mentioned radio switches at the same time.

ACTIVITY MONITOR

nmcli monitor

Observe NetworkManager activity. Watches for changes in connectivity state, devices or connection profiles.

See also **nmcli connection monitor** and **nmcli device monitor** to watch for changes in certain devices or connections.

CONNECTION MANAGEMENT COMMANDS

nmcli connection {show | up | down | modify | add | edit | clone | delete | monitor | reload | load | import | export} [ARGUMENTS...]

NetworkManager stores all network configuration as "connections", which are collections of data (Layer2 details, IP addressing, etc.) that describe how to create or connect to a network. A connection is "active" when a device uses that connection's configuration to create or connect to a network. There may be multiple connections that apply to a device, but only one of them can be active on that device at any given time. The additional connections can be used to allow quick switching between different networks and configurations.

Consider a machine which is usually connected to a DHCP-enabled network, but sometimes connected to a testing network which uses static IP addressing. Instead of manually reconfiguring eth0 each time the network is changed, the settings can be saved as two connections which both apply to eth0, one for DHCP (called default) and one with the static addressing details (called testing). When connected to the DHCP-enabled network the user would run **nmcli con up default**, and when connected to the static network the user would run **nmcli con up testing**.

show [--active] [--order [+--]category:...]

List in-memory and on-disk connection profiles, some of which may also be active if a device is using that connection profile. Without a parameter, all profiles are listed. When **--active** option is specified, only the active profiles are shown.

The **--order** option can be used to get custom ordering of connections. The connections can be ordered by active status (active), name (name), type (type) or D-Bus path (path). If connections are equal according to a sort order category, an additional category can be specified. The default sorting order is equivalent to **--order active:name:path**. + or no prefix means sorting in ascending order (alphabetically or in numbers), - means reverse (descending) order. The category names can be abbreviated (e.g. **--order -a:na**).

show [--active] [id | uuid | path | apath] ID...

Show details for specified connections. By default, both static configuration and active connection data are displayed. When **--active** option is specified, only the active profiles are taken into account. Use global **--show-secrets** option to display secrets associated with the profile.

id, **uuid**, **path** and **apath** keywords can be used if *ID* is ambiguous. Optional *ID*-specifying keywords are:

id

the *ID* denotes a connection name.

uuid

the *ID* denotes a connection UUID.

path

the *ID* denotes a D-Bus static connection path in the format of
`/org/freedesktop/NetworkManager/Settings/num` or just *num*.

apath

the *ID* denotes a D-Bus active connection path in the format of
`/org/freedesktop/NetworkManager/ActiveConnection/num` or just *num*.

It is possible to filter the output using the global **--fields** option. Use the following values:

profile

only shows static profile configuration.

active

only shows active connection data (when the profile is active).

You can also specify particular fields. For static configuration, use setting and property names as described in **nm-settings-nmcli(5)** manual page. For active data use GENERAL, IP4, DHCP4, IP6, DHCP6, VPN.

When no command is given to the **nmcli connection**, the default action is **nmcli connection show**.

up [id | uuid | path] ID [ifname ifname] [ap BSSID] [passwd-file file]

Activate a connection. The connection is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid** or **path** can be used. When requiring a particular device to activate the connection on, the **ifname** option with interface name should be given. If the *ID* is not given an **ifname** is required, and NetworkManager will activate the best available connection for the given **ifname**. In case of a VPN connection, the **ifname** option specifies the device of the base connection. The **ap** option specify what particular AP should be used in case of a Wi-Fi connection.

If **--wait** option is not specified, the default timeout will be 90 seconds.

See **connection show** above for the description of the *ID*-specifying keywords.

Available options are:

ifname

interface that will be used for activation.

ap

BSSID of the AP which the command should connect to (for Wi-Fi connections).

passwd-file

some networks may require credentials during activation. You can give these credentials using this option. Each line of the file should contain one password in the form:

`setting_name.property_name:the password`

For example, for WPA Wi-Fi with PSK, the line would be

```
802-11-wireless-security.psk:secret12345
```

For 802.1X password, the line would be

```
802-1x.password:my 1X password
```

nmcli also accepts wifi-sec and wifi strings instead of 802-11-wireless-security. When NetworkManager requires a password and it is not given, **nmcli** will ask for it when run with --ask. If --ask was not passed, NetworkManager can ask another secret agent that may be running (typically a GUI secret agent, such as nm-applet or gnome-shell).

down [id | uuid | path | apath] ID...

Deactivate a connection from a device without preventing the device from further auto-activation. Multiple connections can be passed to the command.

Be aware that this command deactivates the specified active connection, but the device on which the connection was active, is still ready to connect and will perform auto-activation by looking for a suitable connection that has the 'autoconnect' flag set. Note that the deactivating connection profile is internally blocked from autoconnecting again. Hence it will not autoconnect until reboot or until the user performs an action that unblocks autoconnect, like modifying the profile or explicitly activating it.

In most cases you may want to use **device down** command instead.

The connection is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid**, **path** or **apath** can be used.

See **connection show** above for the description of the *ID*-specifying keywords.

If --wait option is not specified, the default timeout will be 10 seconds.

modify [--temporary] [id | uuid | path] ID {option value | [+|-]setting.property value}...

Add, modify or remove properties in the connection profile.

To set the property just specify the property name followed by the value. An empty value ("") resets the property value to the default.

See **nm-settings-nmcli(5)** for complete reference of setting and property names, their descriptions and default values. The *setting* and *property* can be abbreviated provided they are unique.

If you want to append an item or a flag to the existing value, use + prefix for the property name or alias. If you want to remove items from a container-type or flag property, use - prefix. For certain properties you can also remove elements by specifying the zero-based index(es). The + and - modifiers only have a real effect for properties that support them. These are for example multi-value (container) properties or flags like ipv4.dns, ip4, ipv4.addresses, bond.options, 802-1x.phase1-auth-flags etc.

The connection is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid** or **path** can be used.

modify [--temporary] [id | uuid | path] ID remove setting

Removes a setting from the connection profile.

add [save {yes | no}] {option value | [+|-]setting.property value}...

Create a new connection using specified properties.

You need to describe the newly created connections with the property and value pairs. See **nm-settings-nmcli(5)** for the complete reference. The syntax is the same as of the **nmcli connection modify** command.

To construct a meaningful connection you at the very least need to set the **connection.type** property (or use the **type** alias) to one of known NetworkManager connection types:

- 6lowpan
- 802-11-olpc-mesh (alias olpc-mesh)
- 802-11-wireless (alias wifi)
- 802-3-ethernet (alias ethernet)
- adsl
- bluetooth
- bond
- bond-slave (deprecated for ethernet with master)
- bridge
- bridge-slave (deprecated for ethernet with master)
- cdma
- dummy
- generic
- gsm
- infiniband
- ip-tunnel
- macsec
- macvlan
- olpc-mesh
- ovs-bridge
- ovs-dpdk
- ovs-interface
- ovs-patch
- ovs-port
- pppoe
- team
- team-slave (deprecated for ethernet with master)
- tun
- veth
- vlan
- vpn
- vrf
- vxlan
- wifi-p2p

- wimax
- wireguard
- wpan

The most typical uses are described in the EXAMPLES section.

Aside from the properties and values two special options are accepted:

save

Controls whether the connection should be persistent, i.e. NetworkManager should store it on disk (default: yes).

If a single -- argument is encountered it is ignored. This is for compatibility with older versions on **nmcli**.

edit {[id | uuid | path] ID | [type type] [con-name name]}

Edit an existing connection or add a new one, using an interactive editor.

The existing connection is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid**, or **path** can be used. See **connection show** above for the description of the *ID*-specifying keywords. Not providing an *ID* means that a new connection will be added.

The interactive editor will guide you through the connection editing and allow you to change connection parameters according to your needs by means of a simple menu-driven interface. The editor indicates what settings and properties can be modified and provides in-line help.

Available options:

type

type of the new connection; valid types are the same as for **connection add** command.

con-name

name for the new connection. It can be changed later in the editor.

See also **nm-settings-nmcli(5)** for all NetworkManager settings and property names, and their descriptions; and **nmcli-examples(7)** for sample editor sessions.

clone [--temporary] [id | uuid | path] ID new_name

Clone a connection. The connection to be cloned is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid** or **path** can be used. See **connection show** above for the description of the *ID*-specifying keywords. *new_name* is the name of the new cloned connection. The new connection will be the exact copy except the connection.id (*new_name*) and connection.uuid (generated) properties.

The new connection profile will be saved as persistent unless **--temporary** option is specified, in which case the new profile won't exist after NetworkManager restart.

delete [id | uuid | path] ID...

Delete a configured connection. The connection to be deleted is identified by its name, UUID or D-Bus path. If *ID* is ambiguous, a keyword **id**, **uuid** or **path** can be used. See **connection show** above for the description of the *ID*-specifying keywords.

If **--wait** option is not specified, the default timeout will be 10 seconds.

monitor [id | uuid | path] ID...

Monitor connection profile activity. This command prints a line whenever the specified connection changes. The connection to be monitored is identified by its name, UUID or D-Bus path. If *ID* is

ambiguous, a keyword **id**, **uuid** or **path** can be used. See **connection show** above for the description of the *ID*-specifying keywords.

Monitors all connection profiles in case none is specified. The command terminates when all monitored connections disappear. If you want to monitor connection creation consider using the global monitor with **nmcli monitor** command.

reload

Reload all connection files from disk. NetworkManager does not monitor changes to connection. So you need to use this command in order to tell NetworkManager to re-read the connection profiles from disk when a change was made to them.

load *filename...*

Load/reload one or more connection files from disk. Use this after manually editing a connection file to ensure that NetworkManager is aware of its latest state.

import [**--temporary**] **type** *type* **file** *file*

Import an external/foreign configuration as a NetworkManager connection profile. The type of the input file is specified by **type** option.

Only VPN configurations are supported at the moment. The configuration is imported by NetworkManager VPN plugins. **type** values are the same as for **vpn-type** option in **nmcli connection add**. VPN configurations are imported by VPN plugins. Therefore the proper VPN plugin has to be installed so that **nmcli** could import the data.

The imported connection profile will be saved as persistent unless **--temporary** option is specified, in which case the new profile won't exist after NetworkManager restart.

export [**id** | **uuid** | **path**] *ID* [*file*]

Export a connection.

Only VPN connections are supported at the moment. A proper VPN plugin has to be installed so that **nmcli** could export a connection. If no *file* is provided, the VPN configuration data will be printed to standard output.

DEVICE MANAGEMENT COMMANDS

```
nmcli device {status | show | set | up | connect | reapply | modify | down | disconnect | delete | monitor |
              wifi | llidp} {ARGUMENTS...}
```

Show and manage network interfaces.

status

Print status of devices.

This is the default action if no command is specified to **nmcli device**.

show [*ifname*]

Show detailed information about devices. Without an argument, all devices are examined. To get information for a specific device, the interface name has to be provided.

set [*ifname*] *ifname* [**autoconnect** {yes | no}] [**managed** {yes | no}]

Set device properties.

up *ifname*

Connect the device. NetworkManager will try to find a suitable connection that will be activated. It will also consider connections that are not set to auto connect.

If no compatible connection exists, a new profile with default settings will be created and activated. This differentiates **nmcli connection up ifname "\$DEVICE"** from **nmcli device up "\$DEVICE"**

If **--wait** option is not specified, the default timeout will be 90 seconds.

connect *ifname*

Alias for command **up**. Before version 1.34.0 **up** was not supported.

reapply *ifname*

Attempt to update device with changes to the currently active connection made since it was last applied.

modify *ifname* {*option value* | [+|-]*setting.property value*}...

Modify the settings currently active on the device.

This command lets you do temporary changes to a configuration active on a particular device. The changes are not preserved in the connection profile.

See **nm-settings-nmcli(5)** for the list of available properties. Please note that some properties can't be changed on an already connected device.

down *ifname*...

Disconnect a device and prevent the device from automatically activating further connections without user/manual intervention. Note that disconnecting software devices may mean that the devices will disappear.

If **--wait** option is not specified, the default timeout will be 10 seconds.

disconnect *ifname*...

Alias for command **down**. Before version 1.34.0 **down** was not supported.

delete *ifname*...

Delete a device. The command removes the interface from the system. Note that this only works for software devices like bonds, bridges, teams, etc. Hardware devices (like Ethernet) cannot be deleted by the command.

If **--wait** option is not specified, the default timeout will be 10 seconds.

monitor [*ifname*...]

Monitor device activity. This command prints a line whenever the specified devices change state.

Monitors all devices in case no interface is specified. The monitor terminates when all specified devices disappear. If you want to monitor device addition consider using the global monitor with **nmcli monitor** command.

wifi [list [--rescan | auto | no | yes] [*ifname ifname*] [**bssid** *BSSID*]]

List available Wi-Fi access points. The **ifname** and **bssid** options can be used to list APs for a particular interface or with a specific BSSID, respectively.

By default, **nmcli** ensures that the access point list is no older than 30 seconds and triggers a network scan if necessary. The **--rescan** can be used to either force or disable the scan regardless of how fresh the access point list is.

wifi connect(B)*SSID* [**passw ord** *password*] [**wep–key–type** {key | phrase}] [*ifname ifname*] [**bssid** *BSSID*] [**name** *name*] [**private** {yes | no}] [**hidden** {yes | no}])

Connect to a Wi-Fi network specified by SSID or BSSID. The command finds a matching connection or creates one and then activates it on a device. This is a command-line counterpart of clicking an SSID in a GUI client. If a connection for the network already exists, it is possible to bring up (activate) the existing profile as follows: **nmcli con up id** *name*. Note that only open, WEP and WPA-PSK networks are supported if no previous connection exists. It is also assumed that IP configuration is obtained via DHCP.

If **--wait** option is not specified, the default timeout will be 90 seconds.

Available options are:

password

password for secured networks (WEP or WPA).

wep-key-type

type of WEP secret, either **key** for ASCII/HEX key or **phrase** for passphrase.

ifname

interface that will be used for activation.

bssid

if specified, the created connection will be restricted just for the BSSID.

name

if specified, the connection will use the name (else NM creates a name itself).

private

if set to yes, the connection will only be visible to the user who created it. Otherwise, the connection is system-wide, which is the default.

hidden

set to yes when connecting for the first time to an AP not broadcasting its SSID. Otherwise, the SSID would not be found and the connection attempt would fail.

wifi hotspot[ifname ifname] [con-name name] [ssid SSID] [band {a | bg}] [channel c channel] [password password]

Create a Wi-Fi hotspot. The command creates a hotspot connection profile according to Wi-Fi device capabilities and activates it on the device. The hotspot is secured with WPA if device/driver supports that, otherwise WEP is used. Use **connection down** or **device down** to stop the hotspot.

Parameters of the hotspot can be influenced by the optional parameters:

ifname

what Wi-Fi device is used.

con-name

name of the created hotspot connection profile.

ssid

SSID of the hotspot.

band

Wi-Fi band to use.

channel

Wi-Fi channel to use.

password

password to use for the created hotspot. If not provided, **nmcli** will generate a password. The password is either WPA pre-shared key or WEP key.

Note that **--show-secrets** global option can be used to print the hotspot password. It is useful especially when the password was generated.

wifi rescan [ifname ifname] [ssid SSID...]

Request that NetworkManager immediately re-scan for available access points. NetworkManager scans Wi-Fi networks periodically, but in some cases it can be useful to start scanning manually (e.g. after resuming the computer). By using **ssid**, it is possible to scan for a specific SSID, which is useful for APs with hidden SSIDs. You can provide multiple **ssid** parameters in order to scan more SSIDs.

This command does not show the APs, use **nmcli device wifi list** for that.

wifi show-password [ifname ifname]

Show the details of the active Wi-Fi networks, including the secrets.

lldp [list [ifname ifname]]

Display information about neighboring devices learned through the Link Layer Discovery Protocol (LLDP). The **ifname** option can be used to list neighbors only for a given interface. The protocol must be enabled in the connection settings.

SECRET AGENT

nmcli agent {secret | polkit | all}

Run **nmcli** as a NetworkManager secret agent, or polkit agent.

secret

Register **nmcli** as a NetworkManager secret agent and listen for secret requests. You usually do not need this command, because **nmcli** can handle secrets when connecting to networks. However, you may find the command useful when you use another tool for activating connections and you do not have a secret agent available (like nm-applet).

polkit

Register **nmcli** as a polkit agent for the user session and listen for authorization requests. You do not usually need this command, because **nmcli** can handle polkit actions related to NetworkManager operations (when run with **--ask**). However, you may find the command useful when you want to run a simple text based polkit agent and you do not have an agent of a desktop environment. Note that running this command makes **nmcli** handle all polkit requests, not only NetworkManager related ones, because only one polkit agent can run for the session.

all

Runs **nmcli** as both NetworkManager secret and a polkit agent.

COLORS

Implicit coloring can be disabled by an empty file /etc/terminal-colors.d/nmcli.disable.

See **terminal-colors.d(5)** for more details about colorization configuration. The logical color names supported by **nmcli** are:

connection-activated

A connection that is active.

connection-activating

Connection that is being activated.

connection-disconnecting

Connection that is being disconnected.

connection-external

Connection representing configuration created externally to NetworkManager.

connection-invisible

Connection whose details is the user not permitted to see.

connectivity-full

Connectivity state when Internet is reachable.

connectivity-limited

Connectivity state when only a local network reachable.

connectivity-none

Connectivity state when the network is disconnected.

connectivity-portal

Connectivity state when a captive portal hijacked the connection.

connectivity-unknown

Connectivity state when a connectivity check didn't run.

device-activated

Device that is connected.

device-activating

Device that is being configured.

device-disconnected

Device that is not connected.

device-external

Device configured externally to NetworkManager.

device-firmware-missing

Warning of a missing device firmware.

device-plugin-missing

Warning of a missing device plugin.

device-unavailable

Device that is not available for activation.

device-disabled

Device is disabled by software or hardware kill switch.

manager-running

Notice that the NetworkManager daemon is available.

manager-starting

Notice that the NetworkManager daemon is being initially connected.

manager-stopped

Notice that the NetworkManager daemon is not available.

permission-auth

An action that requires user authentication to get permission.

permission-no

An action that is not permitted.

permission-yes

An action that is permitted.

prompt

Prompt in interactive mode.

state-asleep

Indication that NetworkManager in suspended state.

state-connected-global

Indication that NetworkManager in connected to Internet.

state-connected-local

Indication that NetworkManager in local network.

state-connected-site

Indication that NetworkManager in connected to networks other than Internet.

state-connecting

Indication that NetworkManager is establishing a network connection.

state-disconnected

Indication that NetworkManager is disconnected from a network.

state-disconnecting

Indication that NetworkManager is being disconnected from a network.

wifi-signal-excellent

Wi-Fi network with an excellent signal level.

wifi-signal-fair

Wi-Fi network with a fair signal level.

wifi-signal-good

Wi-Fi network with a good signal level.

wifi-signal-poor

Wi-Fi network with a poor signal level.

wifi-signal-unknown

Wi-Fi network that hasn't been actually seen (a hidden AP).

disabled

A property that is turned off.

enabled

A property that is turned on.

ENVIRONMENT VARIABLES

nmcli's behavior is affected by the following environment variables.

LC_ALL

If set to a non-empty string value, it overrides the values of all the other internationalization variables.

LC_MESSAGES

Determines the locale to be used for internationalized messages.

LANG

Provides a default value for the internationalization variables that are unset or null.

INTERNATIONALIZATION NOTES

Be aware that **nmcli** is localized and that is why the output depends on your environment. This is important to realize especially when you parse the output.

Call **nmcli** as **LC_ALL=C nmcli** to be sure the locale is set to C while executing in a script.

LC_ALL, **LC_MESSAGES**, **LANG** variables specify the **LC_MESSAGES** locale category (in that order), which determines the language that **nmcli** uses for messages. The C locale is used if none of these variables are set, and this locale uses English messages.

EXIT STATUS

nmcli exits with status 0 if it succeeds, a value greater than 0 is returned if an error occurs.

0

Success – indicates the operation succeeded.

1

Unknown or unspecified error.

2

Invalid user input, wrong **nmcli** invocation.

3

Timeout expired (see **--wait** option).

4

Connection activation failed.

5

Connection deactivation failed.

6

Disconnecting device failed.

7

Connection deletion failed.

8

NetworkManager is not running.

10

Connection, device, or access point does not exist.

65

When used with **--complete-args** option, a file name is expected to follow.

EXAMPLES

This section presents various examples of **nmcli** usage. If you want even more, please refer to **nmcli-examples(7)** manual page.

nmcli -t -f RUNNING general

tells you whether NetworkManager is running or not.

nmcli -t -f STATE general

shows the overall status of NetworkManager.

nmcli radio wifi off

switches Wi-Fi off.

nmcli connection show

lists all connections NetworkManager has.

nmcli -p -m multiline -f all con show

shows all configured connections in multi-line mode.

nmcli connection show --active

lists all currently active connections.

nmcli -f name,autoconnect c s

shows all connection profile names and their auto-connect property.

nmcli -p connection show "My default em1"

shows details for "My default em1" connection profile.

nmcli --show-secrets connection show "My Home Wi-Fi"

shows details for "My Home Wi-Fi" connection profile with all passwords. Without **--show-secrets** option, secrets would not be displayed.

nmcli -f active connection show "My default em1"

shows details for "My default em1" active connection, like IP, DHCP information, etc.

nmcli -f profile con s "My wired connection"

shows static configuration details of the connection profile with "My wired connection" name.

nmcli -p con up "My wired connection" ifname eth0

activates the connection profile with name "My wired connection" on interface eth0. The **-p** option makes **nmcli** show progress of the activation.

nmcli con up 6b028a27-6dc9-4411-9886-e9ad1dd43761 ap 00:3A:98:7C:42:D3

connects the Wi-Fi connection with UUID 6b028a27-6dc9-4411-9886-e9ad1dd43761 to the AP with BSSID 00:3A:98:7C:42:D3.

nmcli device status

shows the status for all devices.

nmcli dev down em2

disconnects a connection on interface em2 and marks the device as unavailable for auto-connecting.

As a result, no connection will automatically be activated on the device until the device's 'autoconnect' is set to TRUE or the user manually activates a connection.

nmcli -f GENERAL,WIFI-PROPERTIES dev show wlan0

shows details for wlan0 interface; only GENERAL and WIFI-PROPERTIES sections will be shown.

nmcli -f CONNECTIONS device show wlp3s0

shows all available connection profiles for your Wi-Fi interface wlp3s0.

nmcli dev wifi

lists available Wi-Fi access points known to NetworkManager.

nmcli dev wifi con "Cafe Hotspot 1" password caffeine name "My cafe"

creates a new connection named "My cafe" and then connects it to "Cafe Hotspot 1" SSID using password "caffeine". This is mainly useful when connecting to "Cafe Hotspot 1" for the first time.

Next time, it is better to use **nmcli con up id "My cafe"** so that the existing connection profile can be used and no additional is created.

nmcli -s dev wifi hotspot con-name QuickHotspot

creates a hotspot profile and connects it. Prints the hotspot password the user should use to connect to the hotspot from other devices.

nmcli dev modify em1 ipv4.method shared

starts IPv4 connection sharing using em1 device. The sharing will be active until the device is disconnected.

nmcli dev modify em1 ipv6.address 2001:db8::a:bad:c0de

temporarily adds an IP address to a device. The address will be removed when the same connection is activated again.

nmcli connection add type ethernet autoconnect no ifname eth0

non-interactively adds an Ethernet connection tied to eth0 interface with automatic IP configuration (DHCP), and disables the connection's autoconnect flag.

nmcli c a ifname Maxipes-fik type vlan dev eth0 id 55

non-interactively adds a VLAN connection with ID 55. The connection will use eth0 and the VLAN interface will be named Maxipes-fik.

nmcli c a ifname eth0 type ethernet ipv4.method disabled ipv6.method link-local

non-interactively adds a connection that will use eth0 Ethernet interface and only have an IPv6 link-local address configured.

nmcli connection edit ethernet-em1-2

edits existing "ethernet-em1-2" connection in the interactive editor.

nmcli connection edit type ethernet con-name "yet another Ethernet connection"

adds a new Ethernet connection in the interactive editor.

nmcli con mod ethernet-2 connection.autoconnect no

modifies 'autoconnect' property in the 'connection' setting of 'ethernet-2' connection.

nmcli con mod "Home Wi-Fi" wifi.mtu 1350

modifies 'mtu' property in the 'wifi' setting of 'Home Wi-Fi' connection.

nmcli con mod em1-1 ipv4.method manual ipv4.addr "192.168.1.23/24 192.168.1.1, 10.10.1.5/8, 10.0.0.11"

sets manual addressing and the addresses in em1-1 profile.

nmcli con modify ABC +ipv4.dns 8.8.8.8

appends a Google public DNS server to DNS servers in ABC profile.

nmcli con modify ABC -ipv4.addresses "192.168.100.25/24 192.168.1.1"

removes the specified IP address from (static) profile ABC.

nmcli con import type openvpn file ~/Downloads/frootvpn.ovpn

imports an OpenVPN configuration to NetworkManager.

nmcli con export corp-vpnc /home/joe/corpvpnc.conf

exports NetworkManager VPN profile corp-vpnc as standard Cisco (vpnc) configuration.

NOTES

nmcli accepts abbreviations, as long as they are a unique prefix in the set of possible options. As new options get added, these abbreviations are not guaranteed to stay unique. For scripting and long term compatibility it is therefore strongly advised to spell out the full option names.

BUGS

There are probably some bugs. If you find a bug, please report it to your distribution or upstream at <https://gitlab.freedesktop.org/NetworkManager/NetworkManager>.

SEE ALSO

nmcli-examples(7), nm-settings-nmcli(5), nm-online(1), NetworkManager(8), NetworkManager.conf(5), nm-applet(1), nm-connection-editor(1), terminal-colors.d(5).

NAME

nohup – run a command immune to hangups, with output to a non-tty

SYNOPSIS

nohup *COMMAND* [*ARG*]...
nohup *OPTION*

DESCRIPTION

Run *COMMAND*, ignoring hangup signals.

--help display this help and exit

--version

output version information and exit

If standard input is a terminal, redirect it from an unreadable file. If standard output is a terminal, append output to 'nohup.out' if possible, '\$HOME/nohup.out' otherwise. If standard error is a terminal, redirect it to standard output. To save output to FILE, use 'nohup *COMMAND* > FILE'.

NOTE: your shell may have its own version of nohup, which usually supersedes the version described here. Please refer to your shell's documentation for details about the options it supports.

AUTHOR

Written by Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/nohup>>
or available locally via: info '(coreutils) nohup invocation'

NAME

nslookup – query Internet name servers interactively

SYNOPSIS

nslookup [–option] [name | –] [server]

DESCRIPTION

nslookup is a program to query Internet domain name servers. **nslookup** has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode prints just the name and requested information for a host or domain.

ARGUMENTS

Interactive mode is entered in the following cases:

- when no arguments are given (the default name server is used);
- when the first argument is a hyphen (–) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen. For example, to change the default query type to host information, with an initial timeout of 10 seconds, type:

```
nslookup -query=hinfo -timeout=10
```

The **–version** option causes **nslookup** to print the version number and immediately exit.

INTERACTIVE COMMANDS**host [server]**

This command looks up information for **host** using the current default server or using **server**, if specified. If **host** is an Internet address and the query type is A or PTR, the name of the host is returned. If **host** is a name and does not have a trailing period (.), the search list is used to qualify the name.

To look up a host not in the current domain, append a period to the name.

server domain | lserver domain

These commands change the default server to **domain**; **lserver** uses the initial server to look up information about **domain**, while **server** uses the current default server. If an authoritative answer cannot be found, the names of servers that might have the answer are returned.

- | | |
|---------------|----------------------------------|
| root | This command is not implemented. |
| finger | This command is not implemented. |
| ls | This command is not implemented. |
| view | This command is not implemented. |
| help | This command is not implemented. |
| ? | This command is not implemented. |
| exit | This command exits the program. |

set keyword[=value]

This command is used to change state information that affects the lookups. Valid keywords are:

- | | |
|------------|--|
| all | This keyword prints the current values of the frequently used options to set . Information about the current default server and host is also printed. |
|------------|--|

class=value

This keyword changes the query class to one of:

- IN** the Internet class
- CH** the Chaos class
- HS** the Hesiod class
- ANY** wildcard

The class specifies the protocol group of the information. The default is **IN**; the abbreviation for this keyword is **cl**.

nodebug

This keyword turns on or off the display of the full response packet, and any intermediate response packets, when searching. The default for this keyword is **nodebug**; the abbreviation for this keyword is **[no]deb**.

- nod2** This keyword turns debugging mode on or off. This displays more about what nslookup is doing. The default is **nod2**.

domain=name

This keyword sets the search list to **name**.

nosearch

If the lookup request contains at least one period, but does not end with a trailing period, this keyword appends the domain names in the domain search list to the request until an answer is received. The default is **search**.

port=value

This keyword changes the default TCP/UDP name server port to **value** from its default, port 53. The abbreviation for this keyword is **po**.

querytype=value | type=value

This keyword changes the type of the information query to **value**. The defaults are **A** and then **AAAA**; the abbreviations for these keywords are **q** and **ty**.

Please note that it is only possible to specify one query type. Only the default behavior looks up both when an alternative is not specified.

norecurse

This keyword tells the name server to query other servers if it does not have the information. The default is **recurse**; the abbreviation for this keyword is **[no]rec**.

ndots=number

This keyword sets the number of dots (label separators) in a domain that disables searching. Absolute names always stop searching.

retry=number

This keyword sets the number of retries to **number**.

timeout=number

This keyword changes the initial timeout interval to wait for a reply to **number**, in seconds.

- novc** This keyword indicates that a virtual circuit should always be used when sending requests to the server. **novc** is the default.
- nofail** This keyword tries the next nameserver if a nameserver responds with SERVFAIL or a referral (nofail), or terminates the query (fail) on such a response. The default is **nofail**.

RETURN VALUES

nslookup returns with an exit status of 1 if any query failed, and 0 otherwise.

IDN SUPPORT

If **nslookup** has been built with IDN (internationalized domain name) support, it can accept and display non-ASCII domain names. **nslookup** appropriately converts character encoding of a domain name before sending a request to a DNS server or displaying a reply from the server. To turn off IDN support, define the **IDN_DISABLE** environment variable. IDN support is disabled if the variable is set when **nslookup** runs, or when the standard output is not a tty.

FILES

/etc/resolv.conf

SEE ALSO

dig(1), **host(1)**, **named(8)**.

AUTHOR

Internet Systems Consortium

COPYRIGHT

2022, Internet Systems Consortium

NAME

pam - portable arbitrary map file format

DESCRIPTION

The PAM image format is a lowest common denominator 2 dimensional map format.

It is designed to be used for any of myriad kinds of graphics, but can theoretically be used for any kind of data that is arranged as a two dimensional rectangular array. Actually, from another perspective it can be seen as a format for data arranged as a three dimensional array.

This format does not define the meaning of the data at any particular point in the array. It could be red, green, and blue light intensities such that the array represents a visual image, or it could be the same red, green, and blue components plus a transparency component, or it could contain annual rainfalls for places on the surface of the Earth. Any process that uses the PAM format must further define the format to specify the meanings of the data.

A PAM image describes a two dimensional grid of tuples. The tuples are arranged in rows and columns. The width of the image is the number of columns. The height of the image is the number of rows. All rows are the same width and all columns are the same height. The tuples may have any degree, but all tuples have the same degree. The degree of the tuples is called the depth of the image. Each member of a tuple is called a sample. A sample is an unsigned integer which represents a locus along a scale which starts at zero and ends at a certain maximum value greater than zero called the maxval. The maxval is the same for every sample in the image. The two dimensional array of all the Nth samples of each tuple is called the Nth plane or Nth channel of the image.

Though the format does not assign any meaning to the tuple values, it does include an optional string that describes that meaning. The contents of this string, called the tuple type, are arbitrary from the point of view of the PAM format, but users of the format may assign meaning to it by convention so they can identify their particular implementations of the PAM format.

The Layout

A PAM file consists of a sequence of one or more PAM images. There are no data, delimiters, or padding before, after, or between images.

Each PAM image consists of a header followed immediately by a raster.

Here is an example header:

```
P7
WIDTH 227
HEIGHT 149
DEPTH 3
MAXVAL 255
TUPLTYPE RGB
ENDHDR
```

The header begins with the ASCII characters "P7" followed by newline. This is the magic number.

The header continues with an arbitrary number of lines of ASCII text. Each line ends with and is delimited by a newline character.

Each header line consists of zero or more whitespace-delimited tokens or begins with "#". If it begins with "#" it is a comment and the rest of this specification does not apply to it.

A header line which has zero tokens is valid but has no meaning.

The type of header line is identified by its first token, which is 8 characters or less:

ENDHDR

This is the last line in the header. The header must contain exactly one of these header lines.

HEIGHT

The second token is a decimal number representing the height of the image (number of rows). The header must contain exactly one of these header lines.

WIDTH

The second token is a decimal number representing the width of the image (number of columns). The header must contain exactly one of these header lines.

DEPTH

The second token is a decimal number representing the depth of the image (number of planes or channels). The header must contain exactly one of these header lines.

MAXVAL

The second token is a decimal number representing the maxval of the image. The header must contain exactly one of these header lines.

TUPLTYPE

The header may contain any number of these header lines, including zero. The rest of the line is part of the tuple type. The rest of the line is not tokenized, but the tuple type does not include any white space immediately following **TUPLTYPE** or at the very end of the line. It does not include a newline. If there are multiple **TUPL TYPE** header lines, the tuple type is the concatenation of the values from each of them, separated by a single blank, in the order in which they appear in the header. If there are no **TUPL TYPE** header lines the tuple type is the null string.

The raster consists of each row of the image, in order from top to bottom, consecutive with no delimiter of any kind between, before, or after, rows.

Each row consists of every tuple in the row, in order from left to right, consecutive with no delimiter of any kind between, before, or after, tuples.

Each tuple consists of every sample in the tuple, in order, consecutive with no delimiter of any kind between, before, or after, samples.

Each sample consists of an unsigned integer in pure binary format, with the most significant byte first. The number of bytes is the minimum number of bytes required to represent the maxval of the image.

PAM Used For PNM (PBM, PGM, or PPM) Images

A common use of PAM images is to represent the older and more concrete PBM, PGM, and PPM images.

A PBM image is conventionally represented as a PAM image of depth 1 with maxval 1 where the one sample in each tuple is 0 to represent a black pixel and 1 to represent a white one. The height, width, and raster bear the obvious relationship to those of the PBM image. The tuple type for PBM images represented as

PAM images is conventionally "BLACKANDWHITE".

A PGM image is conventionally represented as a PAM image of depth 1. The maxval, height, width, and raster bear the obvious relationship to those of the PGM image. The tuple type for PGM images represented as PAM images is conventionally "GRAYSCALE".

A PPM image is conventionally represented as a PAM image of depth 3. The maxval, height, width, and raster bear the obvious relationship to those of the PPM image. The first plane represents red, the second green, and the third blue. The tuple type for PPM images represented as PAM images is conventionally "RGB".

The Confusing Universe of Netpbm Formats

It is easy to get confused about the relationship between the PAM format and PBM, PGM, PPM, and PNM. Here is a little enlightenment:

"PNM" is not really a format. It is a shorthand for the PBM, PGM, and PPM formats collectively. It is also the name of a group of library functions that can each handle all three of those formats.

"PAM" is in fact a fourth format. But it is so general that you can represent the same information in a PAM image as you can in a PBM, PGM, or PPM image. And in fact a program that is designed to read PBM, PGM, or PPM and does so with a recent version of the Netpbm library, will read an equivalent PAM image just fine and the program will never know the difference.

To confuse things more, there is a collection of library routines called the "pam" functions that read and write the PAM format, but also read and write the PBM, PGM, and PPM formats. They do this because the latter formats are much older and more popular, so this makes it convenient to write programs that use the newer PAM format.

SEE ALSO

pbm(5), pgm(5), ppm(5), pnm(5), libpnm(3)

NAME

PAM, pam – Pluggable Authentication Modules for Linux

DESCRIPTION

This manual is intended to offer a quick introduction to **Linux–PAM**. For more information the reader is directed to the **Linux–PAM system administrators' guide**.

Linux–PAM is a system of libraries that handle the authentication tasks of applications (services) on the system. The library provides a stable general interface (Application Programming Interface – API) that privilege granting programs (such as **login(1)** and **su(1)**) defer to to perform standard authentication tasks.

The principal feature of the PAM approach is that the nature of the authentication is dynamically configurable. In other words, the system administrator is free to choose how individual service—providing applications will authenticate users. This dynamic configuration is set by the contents of the single **Linux–PAM** configuration file `/etc/pam.conf`. Alternatively, the configuration can be set by individual configuration files located in the `/etc/pam.d/` directory. The presence of this directory will cause **Linux–PAM** to *ignore* `/etc/pam.conf`.

Vendor-supplied PAM configuration files might be installed in the system directory `/usr/lib/pam.d/` or a configurable vendor specific directory instead of the machine configuration directory `/etc/pam.d/`. If no machine configuration file is found, the vendor-supplied file is used. All files in `/etc/pam.d/` override files with the same name in other directories.

From the point of view of the system administrator, for whom this manual is provided, it is not of primary importance to understand the internal behavior of the **Linux–PAM** library. The important point to recognize is that the configuration file(s) *define* the connection between applications (**services**) and the pluggable authentication modules (**PAMs**) that perform the actual authentication tasks.

Linux–PAM separates the tasks of *authentication* into four independent management groups: **account** management; **authentication** management; **password** management; and **session** management. (We highlight the abbreviations used for these groups in the configuration file.)

Simply put, these groups take care of different aspects of a typical user's request for a restricted service:

account – provide account verification types of service: has the user's password expired?; is this user permitted access to the requested service?

authentication – authenticate a user and set up user credentials. Typically this is via some challenge-response request that the user must satisfy: if you are who you claim to be please enter your password. Not all authentications are of this type, there exist hardware based authentication schemes (such as the use of smart-cards and biometric devices), with suitable modules, these may be substituted seamlessly for more standard approaches to authentication – such is the flexibility of **Linux–PAM**.

password – this group's responsibility is the task of updating authentication mechanisms. Typically, such services are strongly coupled to those of the **auth** group. Some authentication mechanisms lend themselves well to being updated with such a function. Standard UN*X password-based access is the obvious example: please enter a replacement password.

session – this group of tasks cover things that should be done prior to a service being given and after it is withdrawn. Such tasks include the maintenance of audit trails and the mounting of the user's home directory. The **session** management group is important as it provides both an opening and closing hook for modules to affect the services available to a user.

FILES

`/etc/pam.conf`

the configuration file

`/etc/pam.d`

the **Linux–PAM** configuration directory. Generally, if this directory is present, the `/etc/pam.conf` file is ignored.

`/usr/lib/pam.d`

the **Linux–PAM** vendor configuration directory. Files in `/etc/pam.d` override files with the same name

in this directory.

<vendordir>/pam.d

the **Linux–PAM** vendor configuration directory. Files in /etc/pam.d and /usr/lib/pam.d override files with the same name in this directory. Only available if Linux–PAM was compiled with vendordir enabled.

ERRORS

Typically errors generated by the **Linux–PAM** system of libraries, will be written to **syslog(3)**.

CONFORMING TO

DCE–RFC 86.0, October 1995. Contains additional features, but remains backwardly compatible with this RFC.

SEE ALSO

pam(3), **pam_authenticate(3)**, **pam_sm_setcred(3)**, **pam_strerror(3)**, **PAM(7)**

NAME

pam.conf, pam.d – PAM configuration files

DESCRIPTION

When a *PAM* aware privilege granting application is started, it activates its attachment to the PAM–API. This activation performs a number of tasks, the most important being the reading of the configuration file(s): /etc/pam.conf. Alternatively, this may be the contents of the /etc/pam.d/ directory. The presence of this directory will cause Linux–PAM to ignore /etc/pam.conf.

These files list the *PAMs* that will do the authentication tasks required by this service, and the appropriate behavior of the PAM–API in the event that individual *PAMs* fail.

The syntax of the /etc/pam.conf configuration file is as follows. The file is made up of a list of rules, each rule is typically placed on a single line, but may be extended with an escaped end of line: ‘\<LF>’. Comments are preceded with ‘#’ marks and extend to the next end of line.

The format of each rule is a space separated collection of tokens, the first three being case–insensitive:

service type control module–path module–arguments

The syntax of files contained in the /etc/pam.d/ directory, are identical except for the absence of any *service* field. In this case, the *service* is the name of the file in the /etc/pam.d/ directory. This filename must be in lower case.

An important feature of *PAM*, is that a number of rules may be *stacked* to combine the services of a number of *PAMs* for a given authentication task.

The *service* is typically the familiar name of the corresponding application: *login* and *su* are good examples. The *service*–name, *other*, is reserved for giving *default* rules. Only lines that mention the current service (or in the absence of such, the *other* entries) will be associated with the given service–application.

The *type* is the management group that the rule corresponds to. It is used to specify which of the management groups the subsequent module is to be associated with. Valid entries are:

account

this module type performs non–authentication based account management. It is typically used to restrict/permit access to a service based on the time of day, currently available system resources (maximum number of users) or perhaps the location of the applicant user — ‘root’ login only on the console.

auth

this module type provides two aspects of authenticating the user. Firstly, it establishes that the user is who they claim to be, by instructing the application to prompt the user for a password or other means of identification. Secondly, the module can grant group membership or other privileges through its credential granting properties.

password

this module type is required for updating the authentication token associated with the user. Typically, there is one module for each ‘challenge/response’ based authentication (auth) type.

session

this module type is associated with doing things that need to be done for the user before/after they can be given service. Such things include the logging of information concerning the opening/closing of some data exchange with a user, mounting directories, etc.

If the *type* value from the list above is prepended with a – character the PAM library will not log to the system log if it is not possible to load the module because it is missing in the system. This can be useful especially for modules which are not always installed on the system and are not required for correct authentication and authorization of the login session.

The third field, *control*, indicates the behavior of the PAM–API should the module fail to succeed in its authentication task. There are two types of syntax for this control field: the simple one has a single simple keyword; the more complicated one involves a square–bracketed selection of *value*=*action* pairs.

For the simple (historical) syntax valid *control* values are:

required

failure of such a PAM will ultimately lead to the PAM-API returning failure but only after the remaining *stacked* modules (for this *service* and *type*) have been invoked.

requisite

like *required*, however, in the case that such a module returns a failure, control is directly returned to the application or to the superior PAM stack. The return value is that associated with the first required or requisite module to fail. Note, this flag can be used to protect against the possibility of a user getting the opportunity to enter a password over an unsafe medium. It is conceivable that such behavior might inform an attacker of valid accounts on a system. This possibility should be weighed against the not insignificant concerns of exposing a sensitive password in a hostile environment.

sufficient

if such a module succeeds and no prior *required* module has failed the PAM framework returns success to the application or to the superior PAM stack immediately without calling any further modules in the stack. A failure of a *sufficient* module is ignored and processing of the PAM module stack continues unaffected.

optional

the success or failure of this module is only important if it is the only module in the stack associated with this *service+type*.

include

include all lines of given type from the configuration file specified as an argument to this control.

substack

include all lines of given type from the configuration file specified as an argument to this control. This differs from *include* in that evaluation of the *done* and *die* actions in a substack does not cause skipping the rest of the complete module stack, but only of the substack. Jumps in a substack also can not make evaluation jump out of it, and the whole substack is counted as one module when the jump is done in a parent stack. The *reset* action will reset the state of a module stack to the state it was in as of beginning of the substack evaluation.

For the more complicated syntax valid *control* values have the following form:

[value1=action1 value2=action2 ...]

Where *valueN* corresponds to the return code from the function invoked in the module for which the line is defined. It is selected from one of these: *success*, *open_err*, *symbol_err*, *service_err*, *system_err*, *buf_err*, *perm_denied*, *auth_err*, *cred_insufficient*, *authinfo_unavail*, *user_unknown*, *maxtries*, *new_authtok_reqd*, *acct_expired*, *session_err*, *cred_unavail*, *cred_expired*, *cred_err*, *no_module_data*, *conv_err*, *authtok_err*, *authtok_recover_err*, *authtok_lock_busy*, *authtok_disable_aging*, *try_again*, *ignore*, *abort*, *authtok_expired*, *module_unknown*, *bad_item*, *conv_again*, *incomplete*, and *default*.

The last of these, *default*, implies 'all *valueN*'s not mentioned explicitly. Note, the full list of PAM errors is available in /usr/include/security/_pam_types.h. The *actionN* can take one of the following forms:

ignore

when used with a stack of modules, the module's return status will not contribute to the return code the application obtains.

bad

this action indicates that the return code should be thought of as indicative of the module failing. If this module is the first in the stack to fail, its status value will be used for that of the whole stack.

die

equivalent to bad with the side effect of terminating the module stack and PAM immediately returning to the application.

ok

this tells PAM that the administrator thinks this return code should contribute directly to the return code of the full stack of modules. In other words, if the former state of the stack would lead to a return of *PAM_SUCCESS*, the module's return code will override this value. Note, if the former state of the stack holds some value that is indicative of a module's failure, this 'ok' value will not be used to override that value.

done

equivalent to ok with the side effect of terminating the module stack and PAM immediately returning to the application.

N (an unsigned integer)

jump over the next N modules in the stack. Note that N equal to 0 is not allowed, it would be treated as *ignore* in such case. The side effect depends on the PAM function call: for *pam_authenticate*, *pam_acct_mgmt*, *pam_chauthtok*, and *pam_open_session* it is *ignore*; for *pam_setcred* and *pam_close_session* it is one of *ignore*, *ok*, or *bad* depending on the module's return value.

reset

clear all memory of the state of the module stack and start again with the next stacked module.

Each of the four keywords: required; requisite; sufficient; and optional, have an equivalent expression in terms of the [...] syntax. They are as follows:

required

[success=ok new_authok_reqd=ok ignore=ignore default=bad]

requisite

[success=ok new_authok_reqd=ok ignore=ignore default=die]

sufficient

[success=done new_authok_reqd=done default=ignore]

optional

[success=ok new_authok_reqd=ok default=ignore]

module-path is either the full filename of the PAM to be used by the application (it begins with a '/'), or a relative pathname from the default module location: /lib/security/ or /lib64/security/, depending on the architecture.

module-arguments are a space separated list of tokens that can be used to modify the specific behavior of the given PAM. Such arguments will be documented for each individual module. Note, if you wish to include spaces in an argument, you should surround that argument with square brackets.

```
squid auth required pam_mysql.so user=passwd_query passwd=mada \
    db=eminence [query=select user_name from internet_service \
    where user_name='%u' and password=PASSWORD('%p') and \
    service='web_proxy']
```

When using this convention, you can include '[' characters inside the string, and if you wish to include a ']' character inside the string that will survive the argument parsing, you should use '\]'. In other words:

[..[.]..] --> ..[.]..

Any line in (one of) the configuration file(s), that is not formatted correctly, will generally tend (erring on the side of caution) to make the authentication process fail. A corresponding error is written to the system log files with a call to **syslog(3)**.

More flexible than the single configuration file is it to configure libpam via the contents of the /etc/pam.d/ directory. In this case the directory is filled with files each of which has a filename equal to a service-name (in lower-case): it is the personal configuration file for the named service.

The syntax of each file in /etc/pam.d/ is similar to that of the /etc/pam.conf file and is made up of lines of the following form:

```
type control module-path module-arguments
```

The only difference being that the service-name is not present. The service-name is of course the name of the given configuration file. For example, /etc/pam.d/login contains the configuration for the **login** service.

SEE ALSO

pam(3), **PAM(8)**, **pam_start(3)**

NAME

pam.conf, pam.d – PAM configuration files

DESCRIPTION

When a *PAM* aware privilege granting application is started, it activates its attachment to the PAM–API. This activation performs a number of tasks, the most important being the reading of the configuration file(s): /etc/pam.conf. Alternatively, this may be the contents of the /etc/pam.d/ directory. The presence of this directory will cause Linux–PAM to ignore /etc/pam.conf.

These files list the *PAMs* that will do the authentication tasks required by this service, and the appropriate behavior of the PAM–API in the event that individual *PAMs* fail.

The syntax of the /etc/pam.conf configuration file is as follows. The file is made up of a list of rules, each rule is typically placed on a single line, but may be extended with an escaped end of line: ‘\<LF>’. Comments are preceded with '#' marks and extend to the next end of line.

The format of each rule is a space separated collection of tokens, the first three being case–insensitive:

service type control module–path module–arguments

The syntax of files contained in the /etc/pam.d/ directory, are identical except for the absence of any *service* field. In this case, the *service* is the name of the file in the /etc/pam.d/ directory. This filename must be in lower case.

An important feature of *PAM*, is that a number of rules may be *stacked* to combine the services of a number of *PAMs* for a given authentication task.

The *service* is typically the familiar name of the corresponding application: *login* and *su* are good examples. The *service*–name, *other*, is reserved for giving *default* rules. Only lines that mention the current service (or in the absence of such, the *other* entries) will be associated with the given service–application.

The *type* is the management group that the rule corresponds to. It is used to specify which of the management groups the subsequent module is to be associated with. Valid entries are:

account

this module type performs non–authentication based account management. It is typically used to restrict/permit access to a service based on the time of day, currently available system resources (maximum number of users) or perhaps the location of the applicant user — 'root' login only on the console.

auth

this module type provides two aspects of authenticating the user. Firstly, it establishes that the user is who they claim to be, by instructing the application to prompt the user for a password or other means of identification. Secondly, the module can grant group membership or other privileges through its credential granting properties.

password

this module type is required for updating the authentication token associated with the user. Typically, there is one module for each 'challenge/response' based authentication (auth) type.

session

this module type is associated with doing things that need to be done for the user before/after they can be given service. Such things include the logging of information concerning the opening/closing of some data exchange with a user, mounting directories, etc.

If the *type* value from the list above is prepended with a – character the PAM library will not log to the system log if it is not possible to load the module because it is missing in the system. This can be useful especially for modules which are not always installed on the system and are not required for correct authentication and authorization of the login session.

The third field, *control*, indicates the behavior of the PAM–API should the module fail to succeed in its authentication task. There are two types of syntax for this control field: the simple one has a single simple keyword; the more complicated one involves a square–bracketed selection of *value*=*action* pairs.

For the simple (historical) syntax valid *control* values are:

required

failure of such a PAM will ultimately lead to the PAM-API returning failure but only after the remaining *stacked* modules (for this *service* and *type*) have been invoked.

requisite

like *required*, however, in the case that such a module returns a failure, control is directly returned to the application or to the superior PAM stack. The return value is that associated with the first required or requisite module to fail. Note, this flag can be used to protect against the possibility of a user getting the opportunity to enter a password over an unsafe medium. It is conceivable that such behavior might inform an attacker of valid accounts on a system. This possibility should be weighed against the not insignificant concerns of exposing a sensitive password in a hostile environment.

sufficient

if such a module succeeds and no prior *required* module has failed the PAM framework returns success to the application or to the superior PAM stack immediately without calling any further modules in the stack. A failure of a *sufficient* module is ignored and processing of the PAM module stack continues unaffected.

optional

the success or failure of this module is only important if it is the only module in the stack associated with this *service+type*.

include

include all lines of given type from the configuration file specified as an argument to this control.

substack

include all lines of given type from the configuration file specified as an argument to this control. This differs from *include* in that evaluation of the *done* and *die* actions in a substack does not cause skipping the rest of the complete module stack, but only of the substack. Jumps in a substack also can not make evaluation jump out of it, and the whole substack is counted as one module when the jump is done in a parent stack. The *reset* action will reset the state of a module stack to the state it was in as of beginning of the substack evaluation.

For the more complicated syntax valid *control* values have the following form:

[value1=action1 value2=action2 ...]

Where *valueN* corresponds to the return code from the function invoked in the module for which the line is defined. It is selected from one of these: *success*, *open_err*, *symbol_err*, *service_err*, *system_err*, *buf_err*, *perm_denied*, *auth_err*, *cred_insufficient*, *authinfo_unavail*, *user_unknown*, *maxtries*, *new_authtok_reqd*, *acct_expired*, *session_err*, *cred_unavail*, *cred_expired*, *cred_err*, *no_module_data*, *conv_err*, *authtok_err*, *authtok_recover_err*, *authtok_lock_busy*, *authtok_disable_aging*, *try_again*, *ignore*, *abort*, *authtok_expired*, *module_unknown*, *bad_item*, *conv_again*, *incomplete*, and *default*.

The last of these, *default*, implies 'all *valueN*'s not mentioned explicitly. Note, the full list of PAM errors is available in /usr/include/security/_pam_types.h. The *actionN* can take one of the following forms:

ignore

when used with a stack of modules, the module's return status will not contribute to the return code the application obtains.

bad

this action indicates that the return code should be thought of as indicative of the module failing. If this module is the first in the stack to fail, its status value will be used for that of the whole stack.

die

equivalent to bad with the side effect of terminating the module stack and PAM immediately returning to the application.

ok

this tells PAM that the administrator thinks this return code should contribute directly to the return code of the full stack of modules. In other words, if the former state of the stack would lead to a return of *PAM_SUCCESS*, the module's return code will override this value. Note, if the former state of the stack holds some value that is indicative of a module's failure, this 'ok' value will not be used to override that value.

done

equivalent to ok with the side effect of terminating the module stack and PAM immediately returning to the application.

N (an unsigned integer)

jump over the next N modules in the stack. Note that N equal to 0 is not allowed, it would be treated as *ignore* in such case. The side effect depends on the PAM function call: for *pam_authenticate*, *pam_acct_mgmt*, *pam_chauthtok*, and *pam_open_session* it is *ignore*; for *pam_setcred* and *pam_close_session* it is one of *ignore*, *ok*, or *bad* depending on the module's return value.

reset

clear all memory of the state of the module stack and start again with the next stacked module.

Each of the four keywords: required; requisite; sufficient; and optional, have an equivalent expression in terms of the [...] syntax. They are as follows:

required

[success=ok new_authok_reqd=ok ignore=ignore default=bad]

requisite

[success=ok new_authok_reqd=ok ignore=ignore default=die]

sufficient

[success=done new_authok_reqd=done default=ignore]

optional

[success=ok new_authok_reqd=ok default=ignore]

module-path is either the full filename of the PAM to be used by the application (it begins with a '/'), or a relative pathname from the default module location: /lib/security/ or /lib64/security/, depending on the architecture.

module-arguments are a space separated list of tokens that can be used to modify the specific behavior of the given PAM. Such arguments will be documented for each individual module. Note, if you wish to include spaces in an argument, you should surround that argument with square brackets.

```
squid auth required pam_mysql.so user=passwd_query passwd=mada \
    db=eminence [query=select user_name from internet_service \
    where user_name='%u' and password=PASSWORD('%p') and \
    service='web_proxy']
```

When using this convention, you can include '[' characters inside the string, and if you wish to include a ']' character inside the string that will survive the argument parsing, you should use '\]'. In other words:

[..[.]..] --> ..[.]..

Any line in (one of) the configuration file(s), that is not formatted correctly, will generally tend (erring on the side of caution) to make the authentication process fail. A corresponding error is written to the system log files with a call to *syslog(3)*.

More flexible than the single configuration file is it to configure libpam via the contents of the /etc/pam.d/ directory. In this case the directory is filled with files each of which has a filename equal to a service-name (in lower-case): it is the personal configuration file for the named service.

The syntax of each file in /etc/pam.d/ is similar to that of the /etc/pam.conf file and is made up of lines of the following form:

```
type control module-path module-arguments
```

The only difference being that the service-name is not present. The service-name is of course the name of the given configuration file. For example, /etc/pam.d/login contains the configuration for the **login** service.

SEE ALSO

pam(3), **PAM(8)**, **pam_start(3)**

NAME

pam_access – PAM module for logdaemon style login access control

SYNOPSIS

pam_access.so [debug] [nodefgroup] [noaudit] [accessfile=*file*] [fieldsep=*sep*] [listsep=*sep*]

DESCRIPTION

The pam_access PAM module is mainly for access management. It provides logdaemon style login access control based on login names, host or domain names, internet addresses or network numbers, or on terminal line names, X \$DISPLAY values, or PAM service names in case of non-networked logins.

By default rules for access management are taken from config file /etc/security/access.conf if you don't specify another file. Then individual *.conf files from the /etc/security/access.d/ directory are read. The files are parsed one after another in the order of the system locale. The effect of the individual files is the same as if all the files were concatenated together in the order of parsing. This means that once a pattern is matched in some file no further files are parsed. If a config file is explicitly specified with the **accessfile** option the files in the above directory are not parsed.

If Linux PAM is compiled with audit support the module will report when it denies access based on origin (host, tty, etc.).

OPTIONS

accessfile=/path/to/access.conf

Indicate an alternative access.conf style configuration file to override the default. This can be useful when different services need different access lists.

debug

A lot of debug information is printed with **syslog(3)**.

noaudit

Do not report logins from disallowed hosts and ttys to the audit subsystem.

fieldsep=separators

This option modifies the field separator character that pam_access will recognize when parsing the access configuration file. For example: **fieldsep=|** will cause the default ':' character to be treated as part of a field value and '|' becomes the field separator. Doing this may be useful in conjunction with a system that wants to use pam_access with X based applications, since the **PAM_TTY** item is likely to be of the form "hostname:0" which includes a ':' character in its value. But you should not need this.

listsep=separators

This option modifies the list separator character that pam_access will recognize when parsing the access configuration file. For example: **listsep=,** will cause the default ' ' (space) and '\t' (tab) characters to be treated as part of a list element value and ',' becomes the only list element separator. Doing this may be useful on a system with group information obtained from a Windows domain, where the default built-in groups "Domain Users", "Domain Admins" contain a space.

nodefgroup

User tokens which are not enclosed in parentheses will not be matched against the group database. The backwards compatible default is to try the group database match even for tokens not enclosed in parentheses.

MODULE TYPES PROVIDED

All module types (**auth**, **account**, **password** and **session**) are provided.

RETURN VALUES

PAM_SUCCESS

Access was granted.

PAM_PERM_DENIED

Access was not granted.

PAM_IGNORE

pam_setcred was called which does nothing.

PAM_ABORT

Not all relevant data or options could be gotten.

PAM_USER_UNKNOWN

The user is not known to the system.

FILES

/etc/security/access.conf

Default configuration file

SEE ALSO

access.conf(5), pam.d(5), pam(7).

AUTHORS

The logdaemon style login access control scheme was designed and implemented by Wietse Venema. The pam_access PAM module was developed by Alexei Nogin <alexei@nogin.dnitz.ru>. The IPv6 support and the network(address) / netmask feature was developed and provided by Mike Becher <mike.becher@lrz-muenchen.de>.

NAME

pam_deny – The locking-out PAM module

SYNOPSIS

pam_deny.so

DESCRIPTION

This module can be used to deny access. It always indicates a failure to the application through the PAM framework. It might be suitable for using for default (the *OTHER*) entries.

OPTIONS

This module does not recognise any options.

MODULE TYPES PROVIDED

All module types (**account**, **auth**, **password** and **session**) are provided.

RETURN VALUES

PAM_AUTH_ERR

This is returned by the account and auth services.

PAM_CRED_ERR

This is returned by the setcred function.

PAM_AUTHTOK_ERR

This is returned by the password service.

PAM_SESSION_ERR

This is returned by the session service.

EXAMPLES

```
#%PAM-1.0
#
# If we don't have config entries for a service, the
# OTHER entries are used. To be secure, warn and deny
# access to everything.
other auth required pam_warn.so
other auth required pam_deny.so
other account required pam_warn.so
other account required pam_deny.so
other password required pam_warn.so
other password required pam_deny.so
other session required pam_warn.so
other session required pam_deny.so
```

SEE ALSO

pam.conf(5), **pam.d(5)**, **pam(7)**

AUTHOR

pam_deny was written by Andrew G. Morgan <morgan@kernel.org>

NAME

pam_env – PAM module to set/unset environment variables

SYNOPSIS

```
pam_env.so [debug] [conffile=conf-file] [envfile=env-file] [readenv=0/1] [user_envfile=env-file]
               [user_readenv=0/1]
```

DESCRIPTION

The pam_env PAM module allows the (un)setting of environment variables. Supported is the use of previously set environment variables as well as *PAM_ITEMS* such as *PAM_RHOST*.

By default rules for (un)setting of variables are taken from the config file /etc/security/pam_env.conf. An alternate file can be specified with the *conffile* option.

Second a file (/etc/environment by default) with simple *KEY=VAL* pairs on separate lines will be read. With the *envfile* option an alternate file can be specified. And with the *readenv* option this can be completely disabled.

Third it will read a user configuration file (\$HOME/.pam_environment by default). The default file can be changed with the *user_envfile* option and it can be turned on and off with the *user_readenv* option.

Since setting of PAM environment variables can have side effects to other modules, this module should be the last one on the stack.

OPTIONS

conffile=/path/to/pam_env.conf

Indicate an alternative pam_env.conf style configuration file to override the default. This can be useful when different services need different environments.

debug

A lot of debug information is printed with **syslog(3)**.

envfile=/path/to/environment

Indicate an alternative environment file to override the default. The syntax are simple *KEY=VAL* pairs on separate lines. The *export* instruction can be specified for bash compatibility, but will be ignored. This can be useful when different services need different environments.

readenv=0/1

Turns on or off the reading of the file specified by envfile (0 is off, 1 is on). By default this option is on.

user_envfile=filename

Indicate an alternative .pam_environment file to override the default. The syntax is the same as for /etc/security/pam_env.conf. The filename is relative to the user home directory. This can be useful when different services need different environments.

user_readenv=0/1

Turns on or off the reading of the user specific environment file. 0 is off, 1 is on. By default this option is off.

MODULE TYPES PROVIDED

The **auth** and **session** module types are provided.

RETURN VALUES

PAM_ABORT

Not all relevant data or options could be gotten.

PAM_BUF_ERR

Memory buffer error.

PAM_IGNORE

No pam_env.conf and environment file was found.

PAM_SUCCESS

Environment variables were set.

FILES

/etc/security/pam_env.conf
Default configuration file

/etc/environment
Default environment file

\$HOME/.pam_environment
User specific environment file

SEE ALSO

pam_env.conf(5), pam.d(5), pam(8), environ(7).

AUTHOR

pam_env was written by Dave Kinchlea <kinch@kinch.ark.com>.

NAME

pam_env.conf, environment – the environment variables config files

DESCRIPTION

The /etc/security/pam_env.conf file specifies the environment variables to be set, unset or modified by **pam_env(8)**. When someone logs in, this file is read and the environment variables are set according.

Each line starts with the variable name, there are then two possible options for each variable DEFAULT and OVERRIDE. DEFAULT allows an administrator to set the value of the variable to some default value, if none is supplied then the empty string is assumed. The OVERRIDE option tells pam_env that it should enter in its value (overriding the default value) if there is one to use. OVERRIDE is not used, "" is assumed and no override will be done.

VARIABLE [DEFAULT=[value]] [OVERRIDE=[value]]

(Possibly non-existent) environment variables may be used in values using the \${string} syntax and (possibly non-existent) PAM_ITEMS as well as HOME and SHELL may be used in values using the @{string} syntax. Both the \$ and @ characters can be backslash escaped to be used as literal values values can be delimited with "", escaped " not supported. Note that many environment variables that you would like to use may not be set by the time the module is called. For example, \${HOME} is used below several times, but many PAM applications don't make it available by the time you need it. The special variables @{HOME} and {@SHELL} are expanded to the values for the user from his *passwd* entry.

The "#" character at start of line (no space at front) can be used to mark this line as a comment line.

The /etc/environment file specifies the environment variables to be set. The file must consist of simple NAME=VALUE pairs on separate lines. The **pam_env(8)** module will read the file after the pam_env.conf file.

EXAMPLES

These are some example lines which might be specified in /etc/security/pam_env.conf.

Set the REMOTEHOST variable for any hosts that are remote, default to "localhost" rather than not being set at all

```
REMOTEHOST    DEFAULT=localhost OVERRIDE=@{PAM_RHOST}
```

Set the DISPLAY variable if it seems reasonable

```
DISPLAY      DEFAULT=${REMOTEHOST}:0.0 OVERRIDE=${DISPLAY}
```

Now some simple variables

```
PAGER        DEFAULT=less
MANPAGER     DEFAULT=less
LESS         DEFAULT="M q e h15 z23 b80"
NNTPSERVER   DEFAULT=localhost
PATH         DEFAULT=${HOME}/bin:/usr/local/bin:/bin\
:/usr/bin:/usr/local/bin/X11:/usr/bin/X11
XDG_DATA_HOME @${HOME}/share/
```

Silly examples of escaped variables, just to show how they work.

```
DOLLAR       DEFAULT=\$ 
DOLLARDOLLAR DEFAULT=    OVERRIDE=\${\$DOLLAR}
DOLLARPLUS   DEFAULT=\${REMOTEHOST}\${REMOTEHOST}
ATSIGN      DEFAULT=""   OVERRIDE=\@
```

SEE ALSO

pam_env(8), pam.d(5), pam(7), environ(7)

AUTHOR

pam_env was written by Dave Kinchlea <kinch@kinch.ark.com>.

NAME

pam_group – PAM module for group access

SYNOPSIS

pam_group.so

DESCRIPTION

The pam_group PAM module does not authenticate the user, but instead it grants group memberships (in the credential setting phase of the authentication module) to the user. Such memberships are based on the service they are applying for.

By default rules for group memberships are taken from config file /etc/security/group.conf.

This module's usefulness relies on the file-systems accessible to the user. The point being that once granted the membership of a group, the user may attempt to create a **setgid** binary with a restricted group ownership. Later, when the user is not given membership to this group, they can recover group membership with the precompiled binary. The reason that the file-systems that the user has access to are so significant, is the fact that when a system is mounted *nosuid* the user is unable to create or execute such a binary file. For this module to provide any level of security, all file-systems that the user has write access to should be mounted *nosuid*.

The pam_group module functions in parallel with the /etc/group file. If the user is granted any groups based on the behavior of this module, they are granted *in addition* to those entries /etc/group (or equivalent).

OPTIONS

This module does not recognise any options.

MODULE TYPES PROVIDED

Only the **auth** module type is provided.

RETURN VALUES

PAM_SUCCESS

group membership was granted.

PAM_ABORT

Not all relevant data could be gotten.

PAM_BUF_ERR

Memory buffer error.

PAM_CRED_ERR

Group membership was not granted.

PAM_IGNORE

pam_sm_authenticate was called which does nothing.

PAM_USER_UNKNOWN

The user is not known to the system.

FILES

/etc/security/group.conf

Default configuration file

SEE ALSO

group.conf(5), pam.d(5), pam(7).

AUTHORS

pam_group was written by Andrew G. Morgan <morgan@kernel.org>.

NAME

parted – a partition manipulation program

SYNOPSIS

parted [options] [device [command [options...]]]

DESCRIPTION

parted is a program to manipulate disk partitions. It supports multiple partition table formats, including MS-DOS and GPT. It is useful for creating space for new operating systems, reorganising disk usage, and copying data to new hard disks.

This manual page documents **parted** briefly. Complete documentation is distributed with the package in GNU Info format.

OPTIONS

-h, --help

displays a help message

-l, --list lists partition layout on all block devices

-m, --machine

displays machine parseable output

-s, --script

never prompts for user intervention

-v, --version

displays the version

-a alignment-type, --align alignment-type

Set alignment for newly created partitions, valid alignment types are:

none Use the minimum alignment allowed by the disk type.

cylinder

Align partitions to cylinders.

minimal

Use minimum alignment as given by the disk topology information. This and the opt value will use layout information provided by the disk to align the logical partition table addresses to actual physical blocks on the disks. The min value is the minimum alignment needed to align the partition properly to physical blocks, which avoids performance degradation.

optimal Use optimum alignment as given by the disk topology information. This aligns to a multiple of the physical block size in a way that guarantees optimal performance.

COMMANDS

[device]

The block device to be used. When none is given, **parted** will use the first block device it finds.

[command [options]]

Specifies the command to be executed. If no command is given, **parted** will present a command prompt. Possible commands are:

help [command]

Print general help, or help on *command* if specified.

align-check type partition

Check if *partition* satisfies the alignment constraint of *type*. *type* must be "minimal" or "optimal".

mklabel *label-type*

Create a new disklabel (partition table) of *label-type*. *label-type* should be one of "aix", "amiga", "bsd", "dvh", "gpt", "loop", "mac", "msdos", "pc98", or "sun".

mkpart [*part-type name fs-type*] *start end*

Create a new partition. *part-type* may be specified only with msdos and dvh partition tables, it should be one of "primary", "logical", or "extended". *name* is required for GPT partition tables and *fs-type* is optional. *fs-type* can be one of "btrfs", "ext2", "ext3", "ext4", "fat16", "fat32", "hfs", "hfs+", "linux-swap", "ntfs", "reiserfs", "udf", or "xfs".

name *partition name*

Set the name of *partition* to *name*. This option works only on Mac, PC98, and GPT disklabels. The name can be placed in double quotes, if necessary. And depending on the shell may need to also be wrapped in single quotes so that the shell doesn't strip off the double quotes.

print

Display the partition table.

quit

Exit from **parted**.

rescue *start end*

Rescue a lost partition that was located somewhere between *start* and *end*. If a partition is found, **parted** will ask if you want to create an entry for it in the partition table.

resizepart *partition end*

Change the *end* position of *partition*. Note that this does not modify any filesystem present in the partition.

rm *partition*

Delete *partition*.

select *device*

Choose *device* as the current device to edit. *device* should usually be a Linux hard disk device, but it can be a partition, software raid device, or an LVM logical volume if necessary.

set *partition flag state*

Change the state of the *flag* on *partition* to *state*. Supported flags are: "boot", "root", "swap", "hidden", "raid", "lvm", "lba", "legacy_boot", "irst", "msftres", "esp", "chromeos_kernel", "bls_boot" and "palo". *state* should be either "on" or "off".

unit *unit*

Set *unit* as the unit to use when displaying locations and sizes, and for interpreting those given by the user when not suffixed with an explicit unit. *unit* can be one of "s" (sectors), "B" (bytes), "kB", "MB", "MiB", "GB", "GiB", "TB", "TiB", "%" (percentage of device size), "cyl" (cylinders), "chs" (cylinders, heads, sectors), or "compact" (megabytes for input, and a human-friendly form for output).

toggle *partition flag*

Toggle the state of *flag* on *partition*.

version

Display version information and a copyright message.

REPORTING BUGS

Report bugs to <bug-parted@gnu.org>

SEE ALSO

fdisk(8), **mkfs(8)**, The *parted* program is fully documented in the **info(1)** format *GNU partitioning software* manual which is distributed with the *parted-doc* Debian package.

AUTHOR

This manual page was written by Timshel Knoll <timshel@debian.org>, for the Debian GNU/Linux system (but may be used by others).

NAME

passwd – change user password

SYNOPSIS

passwd [*options*] [*LOGIN*]

DESCRIPTION

The **passwd** command changes passwords for user accounts. A normal user may only change the password for their own account, while the superuser may change the password for any account. **passwd** also changes the account or associated password validity period.

Password Changes

The user is first prompted for their old password, if one is present. This password is then encrypted and compared against the stored password. The user has only one chance to enter the correct password. The superuser is permitted to bypass this step so that forgotten passwords may be changed.

After the password has been entered, password aging information is checked to see if the user is permitted to change the password at this time. If not, **passwd** refuses to change the password and exits.

The user is then prompted twice for a replacement password. The second entry is compared against the first and both are required to match in order for the password to be changed.

Then, the password is tested for complexity. As a general guideline, passwords should consist of 6 to 8 characters including one or more characters from each of the following sets:

- lower case alphabetics
- digits 0 thru 9
- punctuation marks

Care must be taken not to include the system default erase or kill characters. **passwd** will reject any password which is not suitably complex.

Hints for user passwords

The security of a password depends upon the strength of the encryption algorithm and the size of the key space. The legacy *UNIX* System encryption method is based on the NBS DES algorithm. More recent methods are now recommended (see **ENCRYPT_METHOD**). The size of the key space depends upon the randomness of the password which is selected.

Compromises in password security normally result from careless password selection or handling. For this reason, you should not select a password which appears in a dictionary or which must be written down. The password should also not be a proper name, your license number, birth date, or street address. Any of these may be used as guesses to violate system security.

You can find advice on how to choose a strong password on http://en.wikipedia.org/wiki/Password_strength

OPTIONS

The options which apply to the **passwd** command are:

-a, --all

This option can be used only with **-S** and causes show status for all users.

-d, --delete

Delete a user's password (make it empty). This is a quick way to disable a password for an account. It will set the named account passwordless.

-e, --expire

Immediately expire an account's password. This in effect can force a user to change their password at the user's next login.

-h, --help

Display help message and exit.

-i, --inactive *INACTIVE*

This option is used to disable an account after the password has been expired for a number of days.

After a user account has had an expired password for *INACTIVE* days, the user may no longer sign on to the account.

-k, --keep-tokens

Indicate password change should be performed only for expired authentication tokens (passwords). The user wishes to keep their non-expired tokens as before.

-l, --lock

Lock the password of the named account. This option disables a password by changing it to a value which matches no possible encrypted value (it adds a '!' at the beginning of the password).

Note that this does not disable the account. The user may still be able to login using another authentication token (e.g. an SSH key). To disable the account, administrators should use **usermod --expiredate 1** (this set the account's expire date to Jan 2, 1970).

Users with a locked password are not allowed to change their password.

-n, --mindays MIN_DAYS

Set the minimum number of days between password changes to *MIN_DAYS*. A value of zero for this field indicates that the user may change their password at any time.

-q, --quiet

Quiet mode.

-r, --repository REPOSITORY

change password in *REPOSITORY* repository

-R, --root CHROOT_DIR

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-S, --status

Display account status information. The status information consists of 7 fields. The first field is the user's login name. The second field indicates if the user account has a locked password (L), has no password (NP), or has a usable password (P). The third field gives the date of the last password change. The next four fields are the minimum age, maximum age, warning period, and inactivity period for the password. These ages are expressed in days.

-u, --unlock

Unlock the password of the named account. This option re-enables a password by changing the password back to its previous value (to the value before using the **-l** option).

-w, --warndays WARN_DAYS

Set the number of days of warning before a password change is required. The *WARN_DAYS* option is the number of days prior to the password expiring that a user will be warned that their password is about to expire.

-x, --maxdays MAX_DAYS

Set the maximum number of days a password remains valid. After *MAX_DAYS*, the password is required to be changed.

Passing the number **-1** as *MAX_DAYS* will remove checking a password's validity.

CAVEATS

Password complexity checking may vary from site to site. The user is urged to select a password as complex as he or she feels comfortable with.

Users may not be able to change their password on a system if NIS is enabled and they are not logged into the NIS server.

passwd uses PAM to authenticate users and to change their passwords.

FILES

- /etc/passwd
User account information.
- /etc/shadow
Secure user account information.
- /etc/pam.d/passwd
PAM configuration for **passwd**.

EXIT VALUES

The **passwd** command exits with the following values:

- 0* success
- 1* permission denied
- 2* invalid combination of options
- 3* unexpected failure, nothing done
- 4* unexpected failure, passwd file missing
- 5* passwd file busy, try again
- 6* invalid argument to option

SEE ALSO

chpasswd(8), **passwd(5)**, **shadow(5)**, **usermod(8)**.

NAME

`openssl-passwd`, `passwd` – compute password hashes

SYNOPSIS

```
openssl passwd [-help] [-crypt] [-1] [-apr1] [-aixmd5] [-5] [-6] [-salt string] [-in file] [-stdin]
[-noverify] [-quiet] [-table] [-rand file...] [-writerand file] {password}
```

DESCRIPTION

The **passwd** command computes the hash of a password typed at run-time or the hash of each password in a list. The password list is taken from the named file for option **-in file**, from stdin for option **-stdin**, or from the command line, or from the terminal otherwise.

OPTIONS**-help**

Print out a usage message.

-crypt

Use the **crypt** algorithm (default).

-1

Use the MD5 based BSD password algorithm **1**.

-apr1

Use the **apr1** algorithm (Apache variant of the BSD algorithm).

-aixmd5

Use the **AIX MD5** algorithm (AIX variant of the BSD algorithm).

-5

-6 Use the **SHA256** / **SHA512** based algorithms defined by Ulrich Drepper. See <<https://www.akkadia.org/drepper/SHA-crypt.txt>>.

-salt *string*

Use the specified salt. When reading a password from the terminal, this implies **-noverify**.

-in *file*

Read passwords from *file*.

-stdin

Read passwords from **stdin**.

-noverify

Don't verify when reading a password from the terminal.

-quiet

Don't output warnings when passwords given at the command line are truncated.

-table

In the output list, prepend the cleartext password and a TAB character to each password hash.

-rand *file...*

A file or files containing random data used to seed the random number generator. Multiple files can be specified separated by an OS-dependent character. The separator is ; for MS-W indows, , for OpenVMS, and : for all others.

[-writerand *file*]

Writes random data to the specified *file* upon exit. This can be used with a subsequent**-rand** flag.

EXAMPLES

```
% openssl passwd -crypt -salt xx password
xxj31ZMTZzkVA
```

```
% openssl passwd -1 -salt xxxxxxxx password
$1$xxxxxxxx$UYCIxa628.9qXjpQCjM4a.
```

```
% openssl passwd -apr1 -salt xxxxxxxx password
```

```
$apr1$xxxxxxxxx$dxHfLAs jHkDRmG83UXe8K0  
% openssl passwd -aixmd5 -salt xxxxxxxxx password  
xxxxxxxxx$8Oaipk/GPKhC64w/YVeFD/
```

COPYRIGHT

Copyright 2000–2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the OpenSSL license (the “License”). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <<https://www.openssl.org/source/license.html>>.

NAME

passwd – password file

DESCRIPTION

The */etc/passwd* file is a text file that describes user login accounts for the system. It should have read permission allowed for all users (many utilities, like **ls(1)** use it to map user IDs to usernames), but write access only for the superuser.

In the good old days there was no great problem with this general read permission. Everybody could read the encrypted passwords, but the hardware was too slow to crack a well-chosen password, and moreover the basic assumption used to be that of a friendly user-community. These days many people run some version of the shadow password suite, where */etc/passwd* has an 'x' character in the password field, and the encrypted passwords are in */etc/shadow*, which is readable by the superuser only.

If the encrypted password, whether in */etc/passwd* or in */etc/shadow*, is an empty string, login is allowed without even asking for a password. Note that this functionality may be intentionally disabled in applications, or configurable (for example using the "**nullok**" or "**nonnull**" arguments to **pam_unix(8)**).

If the encrypted password in */etc/passwd* is "*NP*" (without the quotes), the shadow record should be obtained from an NIS+ server.

Regardless of whether shadow passwords are used, many system administrators use an asterisk (*) in the encrypted password field to make sure that this user can not authenticate themselves using a password. (But see NOTES below.)

If you create a new login, first put an asterisk (*) in the password field, then use **passwd(1)** to set it.

Each line of the file describes a single user, and contains seven colon-separated fields:

```
name :password:UID:GECOS:directory:shell
```

The field are as follows:

<i>name</i>	This is the user's login name. It should not contain capital letters.
<i>password</i>	This is either the encrypted user password, an asterisk (*), or the letter 'x'. (See pwconv(8) for an explanation of 'x').
<i>UID</i>	The privileged <i>root</i> login account (superuser) has the user ID 0.
<i>GID</i>	This is the numeric primary group ID for this user. (Additional groups for the user are defined in the system group file; see group(5)).
<i>GECOS</i>	This field (sometimes called the "comment field") is optional and used only for informational purposes. Usually, it contains the full username. Some programs (for example, finger(1)) display information from this field. GECOS stands for "General Electric Comprehensive Operating System", which was renamed to GCOS when GE's large systems division was sold to Honeywell. Dennis Ritchie has reported: "Sometimes we sent printer output or batch jobs to the GCOS machine. The gcos field in the password file was a place to stash the information for the \$IDENTcard. Not elegant."
<i>directory</i>	This is the user's home directory: the initial directory where the user is placed after logging in. The value in this field is used to set the HOME environment variable.
<i>shell</i>	This is the program to run at login (if empty, use <i>/bin/sh</i>). If set to a nonexistent executable, the user will be unable to login through login(1) . The value in this field is used to set the SHELL environment variable.

FILES

/etc/passwd

NOTES

If you want to create user groups, there must be an entry in */etc/group*, or no group will exist.

If the encrypted password is set to an asterisk (*), the user will be unable to login using **login(1)**, but may still login using **rlogin(1)**, run existing processes and initiate new ones through **rsh(1)**, **cron(8)**, **at(1)**, or mail filters, etc. Trying to lock an account by simply changing the shell field yields the same result and additionally allows the use of **su(1)**.

SEE ALSO

chfn(1), **chsh(1)**, **login(1)**, **passwd(1)**, **su(1)**, **crypt(3)**, **getpwent(3)**, **getpwnam(3)**, **group(5)**, **shadow(5)**, **vipw(8)**

NAME

pgrep, **pkill**, **pidwait** – look up, signal, or wait for processes based on name and other attributes

SYNOPSIS

```
pgrep [options] pattern
pkill [options] pattern
pidwait [options] pattern
```

DESCRIPTION

pgrep looks through the currently running processes and lists the process IDs which match the selection criteria to stdout. All the criteria have to match. For example,

```
$ pgrep -u root sshd
```

will only list the processes called **sshd** AND owned by **root**. On the other hand,

```
$ pgrep -u root,daemon
```

will list the processes owned by **root** OR **daemon**.

pkill will send the specified signal (by default **SIGTERM**) to each process instead of listing them on stdout.

pidwait will wait for each process instead of listing them on stdout.

OPTIONS

-signal

--signal *signal*

Defines the signal to send to each matched process. Either the numeric or the symbolic signal name can be used. (**pkill** only.)

-c, --count

Suppress normal output; instead print a count of matching processes. When count does not match anything, e.g. returns zero, the command will return non-zero value. Note that for **pkill** and **pidwait**, the count is the number of matching processes, not the processes that were successfully signaled or waited for.

-d, --delimiter *delimiter*

Sets the string used to delimit each process ID in the output (by default a newline). (**pgrep** only.)

-e, --echo

Display name and PID of the process being killed. (**pkill** only.)

-f, --full

The *pattern* is normally only matched against the process name. When **-f** is set, the full command line is used.

-g, --pgroup *pgrp*...

Only match processes in the process group IDs listed. Process group 0 is translated into **pgrep**'s, **pkill**'s, or **pidwait**'s own process group.

-G, --group *gid*...

Only match processes whose real group ID is listed. Either the numerical or symbolical value may be used.

-i, --ignore-case

Match processes case-insensitively.

-l, --list-name

List the process name as well as the process ID. (**pgrep** only.)

-a, --list-full

List the full command line as well as the process ID. (**pgrep** only.)

- n, --newest**
Select only the newest (most recently started) of the matching processes.
- o, --oldest**
Select only the oldest (least recently started) of the matching processes.
- O, --older secs**
Select processes older than secs.
- P, --parent ppid,...**
Only match processes whose parent process ID is listed.
- s, --session sid,...**
Only match processes whose process session ID is listed. Session ID 0 is translated into **pgrep**'s, **pkill**'s, or **pidwait**'s own session ID.
- t, --terminal term,...**
Only match processes whose controlling terminal is listed. The terminal name should be specified without the "/dev/" prefix.
- u, --euid euid,...**
Only match processes whose effective user ID is listed. Either the numerical or symbolical value may be used.
- U, --uid uid,...**
Only match processes whose real user ID is listed. Either the numerical or symbolical value may be used.
- v, --inverse**
Negates the matching. This option is usually used in **pgrep**'s or **pidwait**'s context. In **pkill**'s context the short option is disabled to avoid accidental usage of the option.
- w, --lightweight**
Shows all thread ids instead of pids in **pgrep**'s or **pidwait**'s context. In **pkill**'s context this option is disabled.
- x, --exact**
Only match processes whose names (or command lines if **-f** is specified) **exactly** match the *pattern*.
- F, --pidfile file**
Read *PIDs* from *file*. This option is more useful for **pkill** or **pid** wait than **pgrep**.
- L, --logpidfile**
Fail if pidfile (see **-F**) not locked.
- r, --runstates D,R,S,Z,...**
Match only processes which match the process state.
- ns pid**
Match processes that belong to the same namespaces. Required to run as root to match processes from other users. See **--nslist** for how to limit which namespaces to match.
- nslist name,...**
Match only the provided namespaces. Available namespaces: ipc, mnt, net, pid, user, uts.
- q, --queue value**
Use **sigqueue(3)** rather than **kill(2)** and the value argument is used to specify an integer to be sent with the signal. If the receiving process has installed a handler for this signal using the SA_INFO flag to **sigaction(2)**, then it can obtain this data via the si_value field of the siginfo_t structure.
- V, --version**
Display version information and exit.

-h, --help

Display help and exit.

OPERANDS

pattern Specifies an Extended Regular Expression for matching against the process names or command lines.

EXAMPLES

Example 1: Find the process ID of the **named** daemon:

```
$ pgrep -u root named
```

Example 2: Make **syslog** reread its configuration file:

```
$ pkill -HUP syslogd
```

Example 3: Give detailed information on all **xterm** processes:

```
$ ps -fp $(pgrep -d, -x xterm)
```

Example 4: Make all **chrome** processes run nicer:

```
$ renice +4 $(pgrep chrome)
```

EXIT STATUS

- 0 One or more processes matched the criteria. For pkill and pidwait, one or more processes must also have been successfully signalled or waited for.
- 1 No processes matched or none of them could be signalled.
- 2 Syntax error in the command line.
- 3 Fatal error: out of memory etc.

NOTES

The process name used for matching is limited to the 15 characters present in the output of /proc/pid/stat. Use the **-f** option to match against the complete command line, /proc/pid/cmdline.

The running **pgrep**, **pkill**, or **pidwait** process will never report itself as a match.

BUGS

The options **-n** and **-o** and **-v** can not be combined. Let me know if you need to do this.

Defunct processes are reported.

SEE ALSO

ps(1), **regex(7)**, **signal(7)**, **sigqueue(3)**, **killall(1)**, **skill(1)**, **kill(1)**, **kill(2)**

AUTHOR

Kjetil Torgrim Homme <kjetilho@ifi.uio.no>

REPORTING BUGS

Please send bug reports to <procps@freelists.org>

NAME

plocate-build – generate index for plocate

SYNOPSIS

plocate-build [OPTION]... MLOCATE_DB PLOCATE_DB

DESCRIPTION

plocate-build creates an index from **plocate(1)** from an index earlier generated by **updatedb(8)** from the **mlocate(1)** package. Most users would rather want to use plocate's own **updatedb(8)**, which is more direct. If *PLOCATE_DB* already exists, it will be overwritten (non-atomically). The file is created such that it is not world-readable, as the final access check is done by **plocate(1)** during the search.

OPTIONS

-b, --block-size SIZE

Create blocks containing *SIZE* filenames each, compress them together, and treat them as the same element in the posting lists. This makes the index smaller (because the compression algorithm gets more context to work with, and because there are fewer elements in each posting list), but also makes posting lists less precise, moving more work to weeding out false positives after posting list intersection.

Making this number larger will make linear search (for **--regex**, or very short patterns) faster, with diminishing returns around 256 filenames per block. However, making it too large will cause more false positives, reducing overall performance for typical queries. The default value is 32. Setting it to 1 makes one (compressed) block per filename.

-p, --plaintext

Treat the input as plain text, with entries delimited by newlines, instead of an mlocate database.

--help Print out usage information, then exit successfully.

--version

Print out version information, then exit successfully.

AUTHOR

Steinar H. Gunderson <steinar+plocate@gunderson.no>

SEE ALSO

plocate(1), **/etc/cron.daily/plocate** (which is called **update-plocate.sh** in the source distribution)

NAME

plocate – find files by name, quickly

SYNOPSIS

plocate [*OPTION*]... *PATTERN*...

DESCRIPTION

plocate finds all files on the system matching the given pattern (or all of the patterns if multiple are given). It does this by means of an index made by **updatedb(8)** or (less commonly) converted from another index by **plocate-build(8)**.

plocate is largely argument-compatible with **mlocate(1)**, but is significantly faster. In particular, it rarely needs to scan through its entire database, unless the pattern is very short (less than three bytes) or you want to search for a regular expression. It does not try to maintain compatibility with BSD locate, or non-UTF-8 filenames and locales. Most I/O is done asynchronously, but the results are synchronized so that output comes in the same order every time.

When multiple patterns are given, **plocate** will search for files that match *all* of them. This is the main incompatibility with **mlocate(1)**, which searches for files that match one or more patterns, unless the **-A** option is given.

By default, patterns are taken to be substrings to search for. If at least one non-escaped globbing metacharacter (*, ?, or []) is given, that pattern is instead taken to be a glob pattern (which means it needs to start and end in * for a substring match). If **--regexp** is given, patterns are instead taken to be (non-anchored) POSIX basic regular expressions, and if **--regex** is given, patterns are taken to be POSIX extended regular expressions. All of this matches **mlocate(1)** behavior.

Like **mlocate(1)**, **plocate** shows all files visible to the calling user (by virtue of having read and execute permissions on all parent directories), and none that are not, by means of running with the setgid bit set to access the index (which is built as root), but by testing visibility as the calling user.

OPTIONS**-A, --all**

Ignored for compatibility with **mlocate(1)**.

-b, --basename

Match only against the file name portion of the path name, ie., the directory names will be excluded from the match (but still printed). This does not speed up the search, but can suppress uninteresting matches.

-c, --count

Do not print each match. Instead, count them, and print out a total number at the end.

-d, --database DBPATH

Find matches in the given database, instead of **/var/lib/plocate/plocate.db**. This argument can be given multiple times, to search multiple databases. It is also possible to give multiple databases in one argument, separated by :. (Any character, including :, and \, can be escaped by prepending a \.)

-e, --existing

Print only entries that refer to files existing at the time **locate** is run. Note that unlike **mlocate(1)**, symlinks are not followed by default (and indeed, there is no option to change this).

-i, --ignore-case

Do a case-insensitive match as given by the current locale (default is case-sensitive, byte-by-byte match). Note that **plocate** does not support the full range of Unicode case folding rules; in particular, searching for β will not give you matches on ss even in a German locale. Also note that this option will be somewhat slower than a case-sensitive match, since it needs to generate more candidates for searching the index.

-p, --ignore-spaces

Ignore punctuation and spaces when matching patterns.

-l, --limit LIMIT

Stop searching after *LIMIT* matches have been found. If **--count** is given, the number printed out will be at most *LIMIT*.

-N, --literal

Print entry names without quoting. Normally, **plocate** will escape special characters in filenames, so that they are safe for consumption by typical shells (similar to the GNU coreutils *shell-escape-always* quoting style), unless printing to a pipe, but this option will turn off such quoting.

-0, --null

Instead of writing a newline after every match, write a NUL (ASCII 0). This is useful for creating unambiguous output when it is to be processed by other tools (like **xargs(1)**), as filenames are allowed to contain embedded newlines.

-r, --regexp

Patterns are taken to be POSIX basic regular expressions. See **egrep(7)** for more information. Note that this forces a linear scan through the entire database, which is slow.

--regex

Like **--regexp**, but patterns are instead taken to be POSIX *extended* regular expressions.

-w, --wholename

Match against the entire path name. This is the default, so unless **-b** is given first (see above), it will not do anything. This option thus exists only as compatibility with **mlocate(1)**.

--help Print out usage information, then exit successfully.**--version**

Print out version information, then exit successfully.

ENVIRONMENT**LOCATE_PATH**

If given, appended after the list of **--database** paths (whether an explicit is given or the default is used). Colon-delimiting and character escaping follows the same rules as for **--database**.

AUTHOR

Steinar H. Gunderson <steinar+plocate@gunderson.no>

SEE ALSO

plocate-build(8), mlocate(1), updatedb(8)

NAME

ps – report a snapshot of the current processes.

SYNOPSIS

ps [*options*]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use **top** instead.

This version of **ps** accepts several kinds of options:

- 1 UNIX options, which may be grouped and must be preceded by a dash.
- 2 BSD options, which may be grouped and must not be used with a dash.
- 3 GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and **ps** implementations that this **ps** is compatible with.

Note that **ps -aux** is distinct from **ps aux**. The POSIX and UNIX standards require that **ps -aux** print all processes owned by a user named *x*, as well as printing all processes that would be selected by the **-a** option. If the user named *x* does not exist, this **ps** may interpret the command as **ps aux** instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

By default, **ps** selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

The use of BSD-style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the **PS_FORMAT** environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (TTYs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so **-M** will be considered identical to **Z** and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.

EXAMPLES

To see every process on the system using standard syntax:

```
ps -e  
ps -ef  
ps -eF  
ps -ely
```

To see every process on the system using BSD syntax:

```
ps ax  
ps axu
```

To print a process tree:

```
ps -ejH  
ps axjf
```

To get info about threads:

```
ps -eLf  
ps axms
```

To get security info:

```
ps -eo euser,ruser,suser,fuser,f,comm,label
ps axZ
ps -eM
```

To see every process running as root (real & effective ID) in user format:

```
ps -U root -u root u
```

To see every process with a user-defined format:

```
ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
ps axo stat,euid,ruid,tty,tpgid,ses,pgroup,ppid,pid,pcpu,comm
ps -Ao pid,tt,user,fnname,tmout,f,wchan
```

Print only the process IDs of syslogd:

```
ps -C syslogd -o pid=
```

Print only the name of PID 42:

```
ps -q 42 -o comm=
```

SIMPLE PROCESS SELECTION

- a** Lift the BSD-style "only yourself" restriction, which is imposed upon the set of all processes when some BSD-style (without "-") options are used or when the **ps** personality setting is BSD-like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes with a terminal (tty), or to list all processes when used together with the **x** option.
- A** Select all processes. Identical to **-e**.
- a** Select all processes except both session leaders (see *getsid(2)*) and processes not associated with a terminal.
- d** Select all processes except session leaders.
- deselect**
Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **-N**.
- e** Select all processes. Identical to **-A**.
- g** Really all, even session leaders. This flag is obsolete and may be discontinued in a future release. It is normally implied by the **a** flag, and is only useful when operating in the sunos4 personality.
- N** Select all processes except those that fulfill the specified conditions (negates the selection). Identical to **--deselect**.
- T** Select all processes associated with this terminal. Identical to the **t** option without any argument.
- r** Restrict the selection to only running processes.
- x** Lift the BSD-style "must have a tty" restriction, which is imposed upon the set of all processes when some BSD-style (without "-") options are used or when the **ps** personality setting is BSD-like. The set of processes selected in this manner is in addition to the set of processes selected by other means. An alternate description is that this option causes **ps** to list all processes owned by you (same EUID as **ps**), or to list all processes when used together with the **a** option.

PROCESS SELECTION BY LIST

These options accept a single argument in the form of a blank-separated or comma-separated list. They can be used multiple times. For example: **ps -p "1 2" -p 3,4**

-123 Identical to **--pid 123**.

123 Identical to **--pid 123**.

-C cmdlist

Select by command name. This selects the processes whose executable name is given in *cmdlist*.
NOTE: The command name is not the same as the command line. Previous versions of procps and

the kernel truncated this command name to 15 characters. This limitation is no longer present in both. If you depended on matching only 15 characters, you may no longer get a match.

-G *grplist*

Select by real group ID (RGID) or name. This selects the processes whose real group name or ID is in the *grplist* list. The real group ID identifies the group of the user who created the process, see *getgid*(2).

-g *grplist*

Select by session OR by effective group name. Selection by session is specified by many standards, but selection by effective group is the logical behavior that several other operating systems use. This **ps** will select by session when the list is completely numeric (as sessions are). Group ID numbers will work only when some group names are also specified. See the **-s** and **--group** options.

--Group *grplist*

Select by real group ID (RGID) or name. Identical to **-G**.

--group *grplist*

Select by effective group ID (EGID) or name. This selects the processes whose effective group name or ID is in *grplist*. The effective group ID describes the group whose file access permissions are used by the process (see *getegid*(2)). The **-g** option is often an alternative to **--group**.

p *pidlist*

Select by process ID. Identical to **-p** and **--pid**.

-p *pidlist*

Select by PID. This selects the processes whose process ID numbers appear in *pidlist*. Identical to **p** and **--pid**.

--pid *pidlist*

Select by process ID. Identical to **-p** and **p**.

--ppid *pidlist*

Select by parent process ID. This selects the processes with a parent process ID in *pidlist*. That is, it selects processes that are children of those listed in *pidlist*.

q *pidlist*

Select by process ID (quick mode). Identical to **-q** and **--quick-pid**.

-q *pidlist*

Select by PID (quick mode). This selects the processes whose process ID numbers appear in *pidlist*. With this option **ps** reads the necessary info only for the pids listed in the *pidlist* and doesn't apply additional filtering rules. The order of pids is unsorted and preserved. No additional selection options, sorting and forest type listings are allowed in this mode. Identical to **q** and **--quick-pid**.

--quick-pid *pidlist*

Select by process ID (quick mode). Identical to **-q** and **q**.

-s *sesslist*

Select by session ID. This selects the processes with a session ID specified in *sesslist*.

--sid *sesslist*

Select by session ID. Identical to **-s**.

t *ttylist* Select by tty. Nearly identical to **-t** and **--tty**, but can also be used with an empty *ttylist* to indicate the terminal associated with **ps**. Using the **T** option is considered cleaner than using **t** with an empty *ttylist*.

-t *ttylist*

Select by tty. This selects the processes associated with the terminals given in *ttylist*. Terminals (ttys, or screens for text output) can be specified in several forms: /dev/ttyS1, ttyS1, S1. A plain

"–" may be used to select processes not attached to any terminal.

--tty *ttylist*

Select by terminal. Identical to **-t** and **t**.

U *userlist*

Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*. The effective user ID describes the user whose file access permissions are used by the process (see *geteuid(2)*). Identical to **u** and **--user**.

-U *userlist*

Select by real user ID (RUID) or name. It selects the processes whose real user name or ID is in the *userlist* list. The real user ID identifies the user who created the process, see *getuid(2)*.

-u *userlist*

Select by effective user ID (EUID) or name. This selects the processes whose effective user name or ID is in *userlist*.

The effective user ID describes the user whose file access permissions are used by the process (see *geteuid(2)*). Identical to **U** and **--user**.

--User *userlist*

Select by real user ID (RUID) or name. Identical to **-U**.

--user *userlist*

Select by effective user ID (EUID) or name. Identical to **-u** and **U**.

OUTPUT FORMAT CONTROL

These options are used to choose the information displayed by **ps**. The output may differ by personality.

-c Show different scheduler information for the **-l** option.

--context

Display security context format (for SELinux).

-f Do full-format listing. This option can be combined with many other UNIX-style options to add additional columns. It also causes the command arguments to be printed. When used with **-L**, the NLWP (number of threads) and LWP (thread ID) columns will be added. See the **c** option, the format keyword **args**, and the format keyword **comm**.

-F Extra full format. See the **-f** option, which **-F** implies.

--format *format*

user-defined format. Identical to **-o** and **o**.

j BSD job control format.

-j Jobs format.

l Display BSD long format.

-l Long format. The **-y** option is often useful with this.

-M Add a column of security data. Identical to **Z** (for SELinux).

O *format*

is preloaded **o** (overloaded). The **BSDO** option can act like **e -O** (user-defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with **-O** or **--sort**). When used as a formatting option, it is identical to **-O**, with the BSD personality.

-O *format*

Like **-o**, but preloaded with some default columns. Identical to **-o pid,format,state,tname,time,command** or **-o pid,format,tname,time,cmd**, see **-o** below.

o *format*

Specify user-defined format. Identical to **-o** and **--format**.

-o *format*

User-defined format. *format* is a single argument in the form of a blank-separated or comma-separated list, which offers a way to specify individual output columns. The recognized keywords are described in the **STANDARD FORMAT SPECIFIERS** section below. Headers may be renamed (**ps -o pid,ruser=RealUser -o comm=Command**) as desired. If all column headers are empty (**ps -o pid= -o comm=**) then the header line will not be output. Column width will increase as needed for wide headers; this may be used to widen up columns such as WCHAN (**ps -o pid,wchan=WIDE-WCHAN-COLUMN -o comm**). Explicit width control (**ps opid, wchan:42,cmd**) is offered too. The behavior of **ps -o pid=X,comm=Y** varies with personality; output may be one column named "X,comm=Y" or two columns named "X" and "Y". Use multiple **-o** options when in doubt. Use the **PS_FORMAT** environment variable to specify a default as desired; DefSysV and DefBSD are macros that may be used to choose the default UNIX or BSD columns.

s Display signal format.

u Display user-oriented format.

v Display virtual memory format.

X Register format.

-y Do not show flags; show rss in place of addr. This option can only be used with **-l**.

Z Add a column of security data. Identical to **-M** (for SELinux).

OUTPUT MODIFIERS

c Show the true command name. This is derived from the name of the executable file, rather than from the argv value. Command arguments and any modifications to them are thus not shown. This option effectively turns the **args** format keyword into the **comm** format keyword; it is useful with the **-f** format option and with the various BSD-style format options, which all normally display the command arguments. See the **-f** option, the format keyword **args**, and the format keyword **comm**.

--cols *n*

Set screen width.

--columns *n*

Set screen width.

--cumulative

Include some dead child process data (as a sum with the parent).

e Show the environment after the command.

f ASCII art process hierarchy (forest).

--forest

ASCII art process tree.

h No header. (or, one header per screen in the BSD personality). The **h** option is problematic.

Standard BSD **ps** uses this option to print a header on each page of output, but older Linux **ps** uses this option to totally disable the header. This version of **ps** follows the Linux usage of not printing the header unless the BSD personality has been selected, in which case it prints a header on each page of output. Regardless of the current personality, you can use the long options **--headers** and **--no-headers** to enable printing headers each page or disable headers entirely, respectively.

-H Show process hierarchy (forest).

--headers

Repeat header lines, one per page of output.

k spec Specify sorting order. Sorting syntax is `[+|-]key[,[+|-]key[,...]]`. Choose a multi-letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to—**sort**.

Examples:

```
ps jaxkuid,-ppid,+pid
ps axk comm o comm,args
ps kstart_time -ef
```

—lines n

Set screen height.

n Numeric output for WCHAN and USER (including all types of UID and GID).

—no-headers

Print no header line at all. —**no-heading** is an alias for this option.

O order

Sorting order (overloaded). The **BSDO** option can act like —**O** (user-defined output format with some common fields predefined) or can be used to specify sort order. Heuristics are used to determine the behavior of this option. To ensure that the desired behavior is obtained (sorting or formatting), specify the option in some other way (e.g. with —**O** or —**sort**).

For sorting, obsolete BSD **O** option syntax is `O[+|-]k1[,[+|-]k2[,...]]`. It orders the processes listing according to the multilevel sort specified by the sequence of one-letter short keys *k1,k2,...* described in the **OBSOLETE SORT KEYS** section below. The "+" is currently optional, merely re-iterating the default direction on a key, but may help to distinguish an **O** sort from an **O** format. The "--" reverses direction only on the key it precedes.

—rows n

Set screen height.

S Sum up some information, such as CPU usage, from dead child processes into their parent. This is useful for examining a system where a parent process repeatedly forks off short-lived children to do work.

—sort spec

Specify sorting order. Sorting syntax is `[+|-]key[,[+|-]key[,...]]`. Choose a multi-letter key from the **STANDARD FORMAT SPECIFIERS** section. The "+" is optional since default direction is increasing numerical or lexicographic order. Identical to **k**. For example: `ps jax --sort=uid,-ppid,+pid`

w Wide output. Use this option twice for unlimited width.

-w Wide output. Use this option twice for unlimited width.

—width n

Set screen width.

THREAD DISPLAY

H Show threads as if they were processes.

-L Show threads, possibly with LWP and NLWP columns.

m Show threads after processes.

-m Show threads after processes.

-T Show threads, possibly with SPID column.

OTHER INFORMATION

—help section

Print a help message. The *section* argument can be one of *simple*, *list*, *output*, *threads*, *misc*, or *all*. The argument can be shortened to one of the underlined letters as in: `s|l|o|t|m|a`.

--info Print debugging info.
L List all format specifiers.
V Print the procps-ng version.
-V Print the procps-ng version.
--version
Print the procps-ng version.

NOTES

This **ps** works by reading the virtual files in /proc. This **ps** does not need to be setuid kmem or have any privileges to run. Do not give this **ps** any special permissions.

CPU usage is currently expressed as the percentage of time spent running during the entire lifetime of a process. This is not ideal, and it does not conform to the standards that **ps** otherwise conforms to. CPU usage is unlikely to add up to exactly 100%.

The SIZE and RSS fields don't count some parts of a process including the page tables, kernel stack, struct thread_info, and struct task_struct. This is usually at least 20 KiB of memory that is always resident. SIZE is the virtual size of the process (code+data+stack).

Processes marked <defunct> are dead processes (so-called "zombies") that remain because their parent has not destroyed them properly. These processes will be destroyed by *init*(8) if the parent process exits.

If the length of the username is greater than the length of the display column, the username will be truncated. See the **-o** and **-O** formatting options to customize length.

Commands options such as **ps -aux** are not recommended as it is a confusion of two different standards. According to the POSIX and UNIX standards, the above command asks to display all processes with a TTY (generally the commands users are running) plus all processes owned by a user named *x*. If that user doesn't exist, then **ps** will assume you really meant **ps aux**.

PROCESS FLAGS

The sum of these values is displayed in the "F" column, which is provided by the **flags** output specifier:

- 1 forked but didn't exec
- 4 used super-user privileges

PROCESS STATE CODES

Here are the different values that the **s**, **stat** and **state** output specifiers (header "STAT" or "S") will display to describe the state of a process:

- D uninterruptible sleep (usually IO)
- I Idle kernel thread
- R running or runnable (on run queue)
- S interruptible sleep (waiting for an event to complete)
- T stopped by job control signal
- t stopped by debugger during the tracing
- W paging (not valid since the 2.6.xx kernel)
- X dead (should never be seen)
- Z defunct ("zombie") process, terminated but not reaped by its parent

For BSD formats and when the **stat** keyword is used, additional characters may be displayed:

- < high-priority (not nice to other users)
- N low-priority (nice to other users)
- L has pages locked into memory (for real-time and custom IO)
- s is a session leader
- l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
- +
- is in the foreground process group

OBSOLETE SORT KEYS

These keys are used by the BSD **O** option (when it is used for sorting). The GNU **--sort** option doesn't use these keys, but the specifiers described below in the **STANDARD FORMAT SPECIFIERS** section. Note that the values used in sorting are the internal values **ps** uses and not the "cooked" values used in some of the output format fields (e.g. sorting on **tty** will sort into device number, not according to the terminal name displayed). Pipe **ps** output into the **sort(1)** command if you want to sort the cooked values.

KEY	LONG	DESCRIPTION
c	cmd	simple name of executable
C	pcpu	cpu utilization
f	flags	flags as in long format F field
g	pgrp	process group ID
G	tpgid	controlling tty process group ID
j	cutime	cumulative user time
J	cstime	cumulative system time
k	utime	user time
m	min_flt	number of minor page faults
M	maj_flt	number of major page faults
n	cmin_flt	cumulative minor page faults
N	cmaj_flt	cumulative major page faults
o	session	session ID
p	pid	process ID
P	ppid	parent process ID
r	rss	resident set size
R	resident	resident pages
s	size	memory size in kilobytes
S	share	amount of shared pages
t	tty	the device number of the controlling tty
T	start_time	time process was started
U	uid	user ID number
u	user	user name
v	vsize	total VM size in KiB
y	priority	kernel scheduling priority

AIX FORMAT DESCRIPTORS

This **ps** supports AIX format descriptors, which work somewhat like the formatting codes of *printf(1)* and *printf(3)*. For example, the normal default output can be produced with this: **ps -eo "%p %y %x %c"**. The **NORMAL** codes are described in the next section.

CODE	NORMAL	HEADER
%C	pcpu	%CPU
%G	group	GROUP
%P	ppid	PPID
%U	user	USER
%a	args	COMMAND
%c	comm	COMMAND
%g	rgroup	RGROUP
%n	nice	NI
%p	pid	PID
%r	pgid	PGID
%t	etime	ELAPSED
%u	ruser	RUSER
%x	time	TIME
%y	tty	TTY
%z	vsz	VSZ

STANDARD FORMAT SPECIFIERS

Here are the different keywords that may be used to control the output format (e.g., with option **-o**) or to sort the selected processes with the GNU-style **--sort** option.

For example: **ps -eo pid,user,args --sort user**

This version of **ps** tries to recognize most of the keywords used in other implementations of **ps**.

The following user-defined format specifiers may contain spaces: **args**, **cmd**, **comm**, **command**, **fname**, **ucmd**, **ucomm**, **lstart**, **bsdstart**, **start**.

Some keywords may not be available for sorting.

CODE	HEADER	DESCRIPTION
%cpu	%CPU	cpu utilization of the process in "##.#" format. Currently, it is the CPU time used divided by the time the process has been running (cputime realtime ratio), expressed as a percentage. It will not add up to 100% unless you are lucky. (alias pcpu).
%mem	%MEM	ratio of the process's resident set size to the physical memory on the machine, expressed as a percentage. (alias pmem).
args	COMMAND	command with all its arguments as a string. Modifications to the arguments may be shown. The output in this column may contain spaces. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. Sometimes the process args will be unavailable; when this happens, ps will instead print the executable name in brackets. (alias cmd , command). See also the comm format keyword, the -f option, and the c option. When specified last, this column will extend to the edge of the display. If ps can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the TERM variable, and so on). The COLUMNS environment variable or --cols option may be used to exactly determine the width in this case. The w or -w option may be also be used to adjust width.
blocked	BLOCKED	mask of the blocked signals, see <i>signal(7)</i> . According to the width of the field, a 32 or 64-bit mask in hexadecimal format is displayed. (alias sig_block , sigmask).
bsdstart	START	time the command started. If the process was started less than 24 hours ago, the output format is " HH:MM", else it is " Mmm:SS" (where Mmm is the three letters of the month). See also lstart , start , start_time , and stime .
bsdtime	TIME	accumulated cpu time, user + system. The display format is usually "MMM:SS", but can be shifted to the right if the process used more than 999 minutes of cpu time.
c	C	processor utilization. Currently, this is the integer value of the percent usage over the lifetime of the process. (see %cpu).
caught	CAUGHT	mask of the caught signals, see <i>signal(7)</i> . According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias sig_catch , sigcatch).

cgroup	CGROUP	display control groups to which the process belongs.																		
class	CLS	scheduling class of the process. (alias policy , cls). Field's possible values are:																		
		<table> <tbody> <tr><td>-</td><td>not reported</td></tr> <tr><td>TS</td><td>SCED_OTHER</td></tr> <tr><td>FF</td><td>SCED_FIFO</td></tr> <tr><td>RR</td><td>SCED_RR</td></tr> <tr><td>B</td><td>SCED_BATCH</td></tr> <tr><td>ISO</td><td>SCED_ISO</td></tr> <tr><td>IDL</td><td>SCED_IDLE</td></tr> <tr><td>DLN</td><td>SCED_DEADLINE</td></tr> <tr><td>?</td><td>unknown value</td></tr> </tbody> </table>	-	not reported	TS	SCED_OTHER	FF	SCED_FIFO	RR	SCED_RR	B	SCED_BATCH	ISO	SCED_ISO	IDL	SCED_IDLE	DLN	SCED_DEADLINE	?	unknown value
-	not reported																			
TS	SCED_OTHER																			
FF	SCED_FIFO																			
RR	SCED_RR																			
B	SCED_BATCH																			
ISO	SCED_ISO																			
IDL	SCED_IDLE																			
DLN	SCED_DEADLINE																			
?	unknown value																			
cls	CLS	scheduling class of the process. (alias policy , cls). Field's possible values are:																		
		<table> <tbody> <tr><td>-</td><td>not reported</td></tr> <tr><td>TS</td><td>SCED_OTHER</td></tr> <tr><td>FF</td><td>SCED_FIFO</td></tr> <tr><td>RR</td><td>SCED_RR</td></tr> <tr><td>B</td><td>SCED_BATCH</td></tr> <tr><td>ISO</td><td>SCED_ISO</td></tr> <tr><td>IDL</td><td>SCED_IDLE</td></tr> <tr><td>DLN</td><td>SCED_DEADLINE</td></tr> <tr><td>?</td><td>unknown value</td></tr> </tbody> </table>	-	not reported	TS	SCED_OTHER	FF	SCED_FIFO	RR	SCED_RR	B	SCED_BATCH	ISO	SCED_ISO	IDL	SCED_IDLE	DLN	SCED_DEADLINE	?	unknown value
-	not reported																			
TS	SCED_OTHER																			
FF	SCED_FIFO																			
RR	SCED_RR																			
B	SCED_BATCH																			
ISO	SCED_ISO																			
IDL	SCED_IDLE																			
DLN	SCED_DEADLINE																			
?	unknown value																			
cmd	CMD	see args . (alias gs , command).																		
comm	COMMAND	command name (only the executable name). Modifications to the command name will not be shown. A process marked <defunct> is partly dead, waiting to be fully destroyed by its parent. The output in this column may contain spaces. (alias ucmd , ucomm). See also the ar gs format keyword, the -f option, and the c option. When specified last, this column will extend to the edge of the display. If ps can not determine display width, as when output is redirected (piped) into a file or another command, the output width is undefined (it may be 80, unlimited, determined by the TERM variable, and so on). The COLUMNS environment variable or --cols option may be used to exactly determine the width in this case. The w or -w option may be also be used to adjust width.																		
command	COMMAND	See args . (alias gs , command).																		
cp	CP	per-mill (tenths of a percent) CPU usage. (see %cpu).																		
cputime	TIME	cumulative CPU time, "[DD-]hh:mm:ss" format. (alias time).																		
cputimes	TIME	cumulative CPU time in seconds (alias times).																		
drs	DRS	data resident set size, the amount of physical memory devoted to other than executable code.																		

egid	EGID	effective group ID number of the process as a decimal integer. (alias gid).
egroup	EGROUP	effective group ID of the process. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias group).
eip	EIP	instruction pointer.
esp	ESP	stack pointer.
etime	ELAPSED	elapsed time since the process was started, in the form [[DD–]hh:]mm:ss.
etimes	ELAPSED	elapsed time since the process was started, in seconds.
euid	EUID	effective user ID (alias uid).
euser	EUSER	effective user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. The n option can be used to force the decimal representation. (alias uname , user).
exe	EXE	path to the executable. Useful if path cannot be printed via cmd , comm or args format options.
f	F	flags associated with the process, see the PROCESS FLAGS section. (alias flag , flags).
fgid	FGID	filesystem access group ID. (alias fsgid).
fgroup	FGROUP	filesystem access group ID. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias fsgroup).
flag	F	see f . (alias f , flags).
flags	F	see f . (alias f , flag).
fname	COMMAND	first 8 bytes of the base name of the process's executable file. The output in this column may contain spaces.
fuid	FUID	filesystem access user ID. (alias fsuid).
fuser	FUSER	filesystem access user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
gid	GID	see egid . (alias egid).
group	GROUP	see egroup . (alias egroup).
ignored	IGNORED	mask of the ignored signals, see <i>signal(7)</i> . According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias sig_ignore , sigignore).

ipcns	IPCNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .
label	LABEL	security label, most commonly used for SELinux context data. This is for the <i>Mandatory Access Control</i> ("MAC") found on high-security systems.
lstart	STARTED	time the command started. See also bsdstart , start , start_time , and stime .
lsession	SESSION	displays the login session identifier of a process, if systemd support has been included.
luid	LUID	displays Login ID associated with a process.
lwp	LWP	light weight process (thread) ID of the dispatchable entity (alias spid , tid). See tid for additional information.
lxc	LXC	The name of the lxc container within which a task is running. If a process is not running inside a container, a dash ('-') will be shown.
machine	MACHINE	displays the machine name for processes assigned to VM or container, if systemd support has been included.
maj_flt	MAJFLT	The number of major page faults that have occurred with this process.
min_flt	MINFLT	The number of minor page faults that have occurred with this process.
mntns	MNTNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .
netns	NETNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .
ni	NI	nice value. This ranges from 19 (nicest) to -20 (not nice to others), see <i>nice(1)</i> . (alias nice).
nice	NI	see ni.(alias ni) .
nlwp	NLWP	number of lwps (threads) in the process. (alias thcount).
numa	NUMA	The node associated with the most recently used processor. A -1 means that NUMA information is unavailable.
nwchan	WCHAN	address of the kernel function where the process is sleeping (use wchan if you want the kernel function name). Running tasks will display a dash ('-') in this column.
ouid	OWNER	displays the Unix user identifier of the owner of the session of a process, if systemd support has been included.
pcpu	%CPU	see %cpu . (alias %cpu).

pending	PENDING	mask of the pending signals. See <i>signal(7)</i> . Signals pending on the process are distinct from signals pending on individual threads. Use the m option or the -m option to see both. According to the width of the field, a 32 or 64 bits mask in hexadecimal format is displayed. (alias sig).																		
pgid	PGID	process group ID or, equivalently, the process ID of the process group leader. (alias pgrp).																		
pgrp	PGRP	see pgid . (alias pgid).																		
pid	PID	a number representing the process ID (alias tgid).																		
pidns	PIDNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .																		
pmem	%MEM	see %mem . (alias %mem).																		
policy	POL	scheduling class of the process. (alias class , cls). Possible values are:																		
		<table> <tr><td>–</td><td>not reported</td></tr> <tr><td>TS</td><td>SCED_OTHER</td></tr> <tr><td>FF</td><td>SCED_FIFO</td></tr> <tr><td>RR</td><td>SCED_RR</td></tr> <tr><td>B</td><td>SCED_BATCH</td></tr> <tr><td>ISO</td><td>SCED_ISO</td></tr> <tr><td>IDL</td><td>SCED_IDLE</td></tr> <tr><td>DLN</td><td>SCED_DEADLINE</td></tr> <tr><td>?</td><td>unknown value</td></tr> </table>	–	not reported	TS	SCED_OTHER	FF	SCED_FIFO	RR	SCED_RR	B	SCED_BATCH	ISO	SCED_ISO	IDL	SCED_IDLE	DLN	SCED_DEADLINE	?	unknown value
–	not reported																			
TS	SCED_OTHER																			
FF	SCED_FIFO																			
RR	SCED_RR																			
B	SCED_BATCH																			
ISO	SCED_ISO																			
IDL	SCED_IDLE																			
DLN	SCED_DEADLINE																			
?	unknown value																			
ppid	PPID	parent process ID.																		
pri	PRI	priority of the process. Higher number means lower priority.																		
psr	PSR	processor that process is currently assigned to.																		
rgid	RGID	real group ID.																		
rgroup	RGROUP	real group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.																		
rss	RSS	resident set size, the non-swapped physical memory that a task has used (in kilobytes). (alias rssize , rsz).																		
rssize	RSS	see rss . (alias rss , rsz).																		
rsz	RSZ	see rss . (alias rss , rssize).																		
rtprio	RTPRIO	realtime priority.																		
ruid	RUID	real user ID.																		
ruser	RUSER	real user ID. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.																		

s	S	minimal state display (one character). See section PROCESS STATE CODES for the different values. See also stat if you want additional information displayed. (alias state).
sched	SCH	scheduling policy of the process. The policies SCHED_OTHER (SCHED_NORMAL), SCHED_FIFO, SCHED_RR, SCHED_BATCH, SCHED_ISO, SCHED_IDLE and SCHED_DEADLINE are respectively displayed as 0, 1, 2, 3, 4, 5 and 6.
seat	SEAT	displays the identifier associated with all hardware devices assigned to a specific workplace, if systemd support has been included.
sess	SESS	session ID or, equivalently, the process ID of the session leader. (alias session , sid).
sgi_p	P	processor that the process is currently executing on. Displays "*" if the process is not currently running or runnable.
sgid	SGID	saved group ID. (alias svgid).
sgroup	SGROUP	saved group name. This will be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.
sid	SID	see sess . (alias session).
sig	PENDING	see pending . (alias pending , sig_pend).
sigcatch	CAUGHT	see caught . (alias caught , sig_catch).
sigignore	IGNORED	see ignored . (alias ignored , sig_ignore).
sigmask	BLOCKED	see blocked . (alias block , sig_block).
size	SIZE	approximate amount of swap space that would be required if the process were to dirty all writable pages and then be swapped out. This number is very rough!
slice	SLICE	displays the slice unit which a process belongs to, if systemd support has been included.
spid	SPID	see lwp . (alias lwp , tid).
stackp	STACKP	address of the bottom (start) of stack for the process.
start	STARTED	time the command started. If the process was started less than 24 hours ago, the output format is "HH:MM:SS", else it is " Mmm dd" (where Mmm is a three-letter month name). See also lstart , bsdstart , start_time , and stime .
start_time	START	starting time or date of the process. Only the year will be displayed if the process was not started the same year ps was invoked, or "MmmDD" if it was not started the same day, or "HH:MM" otherwise. See also bsdstart , start , lstart , and stime .

stat	STAT	multi-character process state. See section PROCESS STATE CODES for the different values meaning. See also s and state if you just want the first character displayed.
state	S	see s . (alias s).
stime	STIME	see start_time . (alias start_time).
suid	SUID	saved user ID. (alias svuid).
supgid	SUPGID	group ids of supplementary groups, if any. See getgr oups(2) .
supgrp	SUPGRP	group names of supplementary groups, if any. See getgr oups(2) .
suser	SUSER	saved user name. This will be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise. (alias svuser).
svgid	SVGID	see sgid . (alias sgid).
svuid	SVUID	see suid . (alias suid).
sz	SZ	size in physical pages of the core image of the process. This includes text, data, and stack space. Device mappings are currently excluded; this is subject to change. See vsz and rss .
tgid	TGID	a number representing the thread group to which a task belongs (alias pid). It is the process ID of the thread group leader.
thcount	THCNT	see nlwp . (alias nlwp). number of kernel threads owned by the process.
tid	TID	the unique number representing a dispatchable entity (alias lwp , spid). This value may also appear as: a process ID (pid); a process group ID (pgrp); a session ID for the session leader (sid); a thread group ID for the thread group leader (tgid); and a tty process group ID for the process group leader (tpgid).
time	TIME	cumulative CPU time, "[DD-]HH:MM:SS" format. (alias cpu_time).
times	TIME	cumulative CPU time in seconds (alias cpu_times).
tname	TTY	controlling tty (terminal). (alias tt , tty).
tpgid	TPGID	ID of the foreground process group on the tty (terminal) that the process is connected to, or -1 if the process is not connected to a tty.
trs	TRS	text resident set size, the amount of physical memory devoted to executable code.
tt	TT	controlling tty (terminal). (alias tname , tty).
tty	TT	controlling tty (terminal). (alias tname , tt).
ucmd	CMD	see comm . (alias comm , ucomm).

ucomm	COMMAND	see comm . (alias comm , ucmd).
uid	UID	see euid . (alias euid).
uname	USER	see euser . (alias euser , user).
unit	UNIT	displays unit which a process belongs to, if systemd support has been included.
user	USER	see euser . (alias euser , uname).
usersns	USERNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .
utsns	UTSNS	Unique inode number describing the namespace the process belongs to. See <i>namespaces(7)</i> .
uunit	UUNIT	displays user unit which a process belongs to, if systemd support has been included.
vsize	VSZ	see vsz . (alias vsz).
vsz	VSZ	virtual memory size of the process in KiB (1024–byte units). Device mappings are currently excluded; this is subject to change. (alias vsize).
wchan	WCHAN	name of the kernel function in which the process is sleeping, a “–” if the process is running, or a “*” if the process is multi-threaded and ps is not displaying threads.

ENVIRONMENT VARIABLES

The following environment variables could affect **ps**:

COLUMNS

Override default display width.

LINES

Override default display height.

PS_PERSONALITY

Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

CMD_ENV

Set to one of posix, old, linux, bsd, sun, digital... (see section **PERSONALITY** below).

I_WANT_A_BROKEN_PS

Force obsolete command line interpretation.

LC_TIME

Date format.

PS_COLORS

Not currently supported.

PS_FORMAT

Default output format override. You may set this to a format string of the type used for the **-o** option.

The **DefSysV** and **DefBSD** values are particularly useful.

POSIXLY_CORRECT

Don't find excuses to ignore bad "features".

POSIX2

When set to "on", acts as **POSIXLY_CORRECT**.

UNIX95

Don't find excuses to ignore bad "features".

XPG

Cancel **CMD_ENV=irix** non-standard behavior.

In general, it is a bad idea to set these variables. The one exception is **CMD_ENV** or **PS_PERSONALITY**, which could be set to Linux for normal systems. Without that setting, **ps** follows the useless and bad parts of the Unix98 standard.

PERSONALITY

390	like the OS/390 OpenEdition ps
aix	like AIX ps
bsd	like FreeBSD ps (totally non-standard)
compaq	like Digital Unix ps
debian	like the old Debian ps
digital	like Tru64 (was Digital Unix, was OSF/1) ps
gnu	like the old Debian ps
hp	like HP-UX ps
hpx	like HP-UX ps
irix	like Irix ps
linux	***** recommended *****
old	like the original Linux ps (totally non-standard)
os390	like OS/390 Open Edition ps
posix	standard
s390	like OS/390 Open Edition ps
sco	like SCO ps
sgi	like Irix ps
solaris2	like Solaris 2+ (SunOS 5) ps
sunos4	like SunOS 4 (Solaris 1) ps (totally non-standard)
svr4	standard
sysv	standard
tru64	like Tru64 (was Digital Unix, was OSF/1) ps
unix	standard
unix95	standard
unix98	standard

SEE ALSO

pgrep(1), pstree(1), top(1), proc(5).

STANDARDS

This **ps** conforms to:

- 1 Version 2 of the Single Unix Specification
- 2 The Open Group Technical Standard Base Specifications, Issue 6
- 3 IEEE Std 1003.1, 2004 Edition
- 4 X/Open System Interfaces Extension [UP XSI]
- 5 ISO/IEC 9945:2003

AUTHOR

ps was originally written by Branko Lankester <lankeste@fwi.uva.nl>. Michael K. Johnson<johnsonm@redhat.com> re-wrote it significantly to use the proc filesystem, changing a few things in the process. Michael Shields <mjshield@nyx.cs.du.edu> added the pid-list feature. Charles Blake <cblake@bbn.com> added multi-level sorting, the dirent-style library, the device name-to-number mmaped database, the

approximate binary search directly on System.map, and many code and documentation cleanups. David Mossberger-Tang wrote the generic BFD support for psupdate. Albert Cahalan <albert@users.sf.net> rewrote ps for full Unix98 and BSD support, along with some ugly hacks for obsolete and foreign syntax.

Please send bug reports to <procps@freelists.org>. No subscription is required or suggested.

NAME

pval – print BER values in ASN.1 value notation

SYNOPSIS

```
pval -T <tt file name> [-m <module name>] -n <type name>  
<BER value file list>
```

DESCRIPTION

pval prints the given BER values in their value notation. You must specify the type name and optionally the module name of the type in the given BER files.

OPTIONS

-T *file*Use the type table in the file to look for the named types definition.

-m *modulename*Specifies the module in which the named type is defined. If the module name is not specified with this option, pval looks for the first occurrence of the named type in the modules in the given type table.

-n *typename*Specifies the type of the values in the given BER files. If you give the wrong type name, decoding errors will occur.

FILES

snacc/tbl-tools/pval/ Source code for the pval program

COPYING

Copyright (c) 1993 Mike Sample and the University of British Columbia

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

AUTHOR

Mike Sample <msample@cs.ubc.ca>, University of British Columbia

ACKNOWLEDGEMENTS

This work was made possible by grants from the Canadian Institute for Telecommunications Research (CITR) and Natural Sciences and Engineering Research Council of Canada (NSERC).

NAME

posix_memalign, **aligned_alloc**, **memalign**, **valloc**, **pvalloc** – allocate aligned memory

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <stdlib.h>
int posix_memalign(void **memptr, size_t alignment, size_t size);
void *aligned_alloc(size_t alignment, size_t size);
void *valloc(size_t size);

#include <malloc.h>
void *memalign(size_t alignment, size_t size);
void *pvalloc(size_t size);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros(7)**):

```
posix_memalign():
    _POSIX_C_SOURCE >= 200112L

aligned_alloc():
    _ISOC11_SOURCE

valloc():
    Since glibc 2.12:
        (_XOPEN_SOURCE >= 500) && !(_POSIX_C_SOURCE >= 200112L)
        || /* glibc >= 2.19: */ _DEFAULT_SOURCE
        || /* glibc <= 2.19: */ _SVID_SOURCE || _BSD_SOURCE

Before glibc 2.12:
    _BSD_SOURCE || _XOPEN_SOURCE >= 500
```

DESCRIPTION

The function **posix_memalign()** allocates *size* bytes and places the address of the allocated memory in **memptr*. The address of the allocated memory will be a multiple of *alignment*, which must be a power of two and a multiple of *sizeof(void *)*. This address can later be successfully passed to **free(3)**. If *size* is 0, then the value placed in **memptr* is either NULL or a unique pointer value.

The obsolete function **memalign()** allocates *size* bytes and returns a pointer to the allocated memory. The memory address will be a multiple of *alignment*, which must be a power of two.

The function **aligned_alloc()** is the same as **memalign()**, except for the added restriction that *size* should be a multiple of *alignment*.

The obsolete function **valloc()** allocates *size* bytes and returns a pointer to the allocated memory. The memory address will be a multiple of the page size. It is equivalent to **memalign(sysconf(_SC_PAGESIZE), size)**.

The obsolete function **pvalloc()** is similar to **valloc()**, but rounds the size of the allocation up to the next multiple of the system page size.

For all of these functions, the memory is not zeroed.

RETURN VALUE

aligned_alloc(), **memalign()**, **valloc()**, and **pvalloc()** return a pointer to the allocated memory on success. On error, NULL is returned, and *errno* is set to indicate the error.

posix_memalign() returns zero on success, or one of the error values listed in the next section on failure. The value of *errno* is not set. On Linux (and other systems), **posix_memalign()** does not modify *memptr* on failure. A requirement standardizing this behavior was added in POSIX.1-2008 TC2.

ERRORS

EINVAL

The *alignment* argument was not a power of two, or was not a multiple of *sizeof(void *)*.

ENOMEM

There was insufficient memory to fulfill the allocation request.

VERSIONS

The functions **memalign()**, **valloc()**, and **pvalloc()** have been available since at least glibc 2.0.

The function **aligned_alloc()** was added in glibc 2.16.

The function **posix_memalign()** is available since glibc 2.1.91.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
aligned_alloc() , memalign() , posix_memalign()	Thread safety	MT-Safe
valloc() , pvalloc()	Thread safety	MT-Unsafe init

STANDARDS

The function **valloc()** appeared in 3.0BSD. It is documented as being obsolete in 4.3BSD, and as legacy in SUSv2. It does not appear in POSIX.1.

The function **pvalloc()** is a GNU extension.

The function **memalign()** appears in SunOS 4.1.3 but not in 4.4BSD.

The function **posix_memalign()** comes from POSIX.1d and is specified in POSIX.1-2001 and POSIX.1-2008.

The function **aligned_alloc()** is specified in the C11 standard.

Headers

Everybody agrees that **posix_memalign()** is declared in `<stdlib.h>`.

On some systems **memalign()** is declared in `<stdlib.h>` instead of `<malloc.h>`.

According to SUSv2, **valloc()** is declared in `<stdlib.h>`. glibc declares it in `<malloc.h>`, and also in `<stdlib.h>` if suitable feature test macros are defined (see above).

NOTES

On many systems there are alignment restrictions, for example, on buffers used for direct block device I/O. POSIX specifies the *pathconf(path, _PC_REC_XFER_ALIGN)* call that tells what alignment is needed. Now one can use **posix_memalign()** to satisfy this requirement.

posix_memalign() verifies that *alignment* matches the requirements detailed above. **memalign()** may not check that the *alignment* argument is correct.

POSIX requires that memory obtained from **posix_memalign()** can be freed using **free(3)**. Some systems provide no way to reclaim memory allocated with **memalign()** or **valloc()** (because one can pass to **free(3)** only a pointer obtained from **malloc(3)**, while, for example, **memalign()** would call **malloc(3)** and then align the obtained value). The glibc implementation allows memory obtained from any of these functions to be reclaimed with **free(3)**.

The glibc **malloc(3)** always returns 8-byte aligned memory addresses, so these functions are needed only if you require larger alignment values.

SEE ALSO

brk(2), **getpagesize(2)**, **free(3)**, **malloc(3)**

NAME

pvchange – Change attributes of physical volume(s)

SYNOPSIS

```
pvchange option_args position_args
      [ option_args ]
```

DESCRIPTION

pvchange changes PV attributes in the VG.

For options listed in parentheses, any one is required, after which the others are optional.

USAGE

Change properties of all PVs.

```
pvchange -a|--all
      ( -x|--allocatable y|n,
        -u|--uuid,
        --addtag Tag,
        --deltag Tag,
        --metadataignore y|n )
      [ COMMON_OPTIONS ]
```

Change properties of specified PVs.

```
pvchange
      ( -x|--allocatable y|n,
        -u|--uuid,
        --addtag Tag,
        --deltag Tag,
        --metadataignore y|n )
      PV|Select ...
      [ -S|--select String ]
      [ COMMON_OPTIONS ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ -u|--uuid ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--addtag *Tag*

Adds a tag to a PV, VG or LV. This option can be repeated to add multiple tags at once. See **lvm(8)** for information about tags.

-a|--all

Change all visible PVs.

-x|--allocatable *y|n*

Enable or disable allocation of physical extents on this PV.

-A|--autobackup *y|n*

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--deletag *Tag*

Deletes a tag from a PV, VG or LV. This option can be repeated to delete multiple tags at once. See **lvm(8)** for information about tags.

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--metadataignore *y|n*

Specifies the metadataignore property of a PV. If yes, metadata areas on the PV are ignored, and lvm will not store metadata in the metadata areas of the PV. If no, lvm will store metadata on the PV.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one

report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select *String*

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by **--select help** and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See **--options help** for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-u|--uuid

Generate new random UUID for specified PVs.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

VARIABLES

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Disallow the allocation of physical extents on a PV (e.g. because of disk errors, or because it will be removed after freeing it).

pvchange -x n /dev/sdk1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgecfgbackup(8) vgecfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvck – Check metadata on physical volumes

SYNOPSIS

```
pvck option_args position_args
      [ option_args ]
      --commandprofile String
      --config String
      -d|--debug
      --driverloaded y|n
      --dump headers|metadata|metadata_all|metadata_search
      -f|--file String
      -h|--help
      --labelsector Number
      --lockopt String
      --longhelp
      --nolocking
      --profile String
      --[pv]metadatacopies 0|1|2
      -q|--quiet
      --repair
      --repairtype pv_header|metadata|label_header
      --settings String
      -t|--test
      -v|--verbose
      --version
      -y|--yes
```

DESCRIPTION

pvck checks and repairs LVM metadata on PVs.

Dump options**headers**

Print LVM on-disk headers and structures: label_header, pv_header, mda_header(s), and metadata text. Warnings are printed if any values are incorrect. The label_header and pv_header both exist in a 512 byte sector, usually the second sector of the device. An mda_header exists in a 512 byte sector at offset 4096 bytes. A second mda_header can optionally exist near the end of the device. The metadata text exists in an area (about 1MiB by default) immediately following the mda_header sector. The metadata text is checked but not printed (see other options).

metadata

Print the current LVM VG metadata text (or save to a file), using headers to locate the latest copy of metadata. If headers are damaged, metadata may not be found (see metadata_search). Use --settings "mda_num=2" to look in mda2 (the second mda at the end of the device, if used). The metadata text is printed to stdout or saved to a file with --file.

metadata_all

List all versions of VG metadata found in the metadata area, using headers to locate metadata. Full copies of all metadata are saved to a file with the --file option. If headers are damaged, metadata may not be found (see metadata_search). Use --settings "mda_num=2" as above. Use -v to include descriptions and dates when listing metadata versions.

metadata_search

List all versions of VG metadata found in the metadata area, searching common locations so metadata can be found if headers are damaged. Full copies of all metadata are saved to a file with the --file option. To

save one specific version of metadata, use `--settings "metadata_offset=<offset>"`, where the offset is taken from the list of versions found. Use `-v` to include descriptions and dates when listing metadata versions.

metadata_area

Save the entire text metadata area to a file without processing.

Repair options**--repair**

Repair headers and metadata on a PV. This uses a metadata input file that was extracted by `--dump`, or a backup file (from `/etc/lvm/backup`). When possible, use metadata saved by `--dump` from another PV in the same VG (or from a second metadata area on the PV).

There are cases where the PV UUID needs to be specified for the PV being repaired. It is specified using `--settings "pv_uuid=<UUID>"`. In particular, if the device name for the PV being repaired does not match the previous device name of the PV, then LVM may not be able to determine the correct PV UUID. When headers are damaged on more than one PV in a VG, it is important for the user to determine the correct PV UUID and specify it in `--settings`. Otherwise, the wrong PV UUID could be used if device names have been swapped since the metadata was last written.

If a PV has no metadata areas and the `pv_header` is damaged, then the repair will not know to create no metadata areas during repair. It will by default repair metadata in `mda1`. To repair with no metadata areas, use `--settings "mda_offset=0 mda_size=0"`.

There are cases where repair should be run on all PVs in the VG (using the same metadata file): if all PVs in the VG are damaged, if using an old metadata version, or if a backup file is used instead of raw metadata (taken from `pvck dump`.)

Using `--repair` is equivalent to running `--repairtype pv_header` followed by `--repairtype metadata`.

--repairtype pv_header

Repairs the header sector, containing the `pv_header` and `label_header`.

--repairtype metadata

Repairs the `mda_header` and metadata text. It requires the headers to be correct (having been undamaged or already repaired).

--repairtype label_header

Repairs `label_header` fields, leaving the `pv_header` (in the same sector) unchanged. (`repairtype pv_header` should usually be used instead.)

Settings

The `--settings` option controls or overrides certain dump or repair behaviors. All offset and size values in settings are in bytes (units are not recognized.) These settings are subject to change.

mda_num=1|2

Select which metadata area should be used. By default the first metadata area (1) is used. `mda1` is always located at offset 4096. `mda2`, at the end of the device, often does not exist (it's not created by default.) If `mda1` is erased, `mda2`, if it exists, will often still have metadata.

metadata_offset=bytes

Select metadata text at this offset. Use with `metadata_search` to print/save one instance of metadata text.

mda_offset=bytes mda_size=bytes

Refers to a metadata area (mda) location and size. An mda includes an `mda_header` and circular metadata

text buffer. Setting this forces metadata_search look for metadata in the given area instead of the standard locations. When set to zero with repair, it indicates no metadata areas should exist.

mda2_offset=bytes mda2_size=bytes

When repairing a pv_header, this forces a specific offset and size for mda2 that should be recorded in the pv_header.

pv_uuid=uuid

Specify the PV UUID of the device being repaired. When not specified, repair will attempt to determine the correct PV UUID by matching a device name in the metadata.

device_size=bytes

data_offset=bytes

When repairing a pv_header, the device_size, data_offset, and pvid can all be specified directly, in which case these values are not taken from a metadata file (where they usually come from), and the metadata file can be omitted. data_offset is the starting location of the first physical extent (data), which follows the first metadata area.

USAGE

Check for metadata on a device

```
pvck PV ...
    [ COMMON_OPTIONS ]
```

Check and print LVM headers and metadata on a device

```
pvck --dump headers|metadata|metadata_all|metadata_search PV
    [ -f|--file String ]
    [ --settings String ]
    [ --[pv]metadatacopies 0|1|2 ]
    [ COMMON_OPTIONS ]
```

Repair LVM headers or metadata on a device

```
pvck --repairtype pv_header|metadata|label_header PV
    [ -f|--file String ]
    [ --settings String ]
    [ COMMON_OPTIONS ]
```

Repair LVM headers and metadata on a device

```
pvck --repair -f|--file String PV
    [ --settings String ]
    [ COMMON_OPTIONS ]
```

Common options for command:

```
[ --labelsector Number ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
```

```
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--dump headers|metadata|metadata_all|metadata_search

Dump headers and metadata from a PV for debugging and repair. Option values include: **headers** to print and check LVM headers, **metadata** to print or save the current text metadata, **meta-data_all** to list or save all versions of metadata, **metadata_search** to list or save all versions of metadata, searching standard locations in case of damaged headers, **metadata_area** to save an entire text metadata area to a file.

-f|--file *String*

Metadata file to read or write.

-h|--help

Display help text.

--labelsector *Number*

By default the PV is labelled with an LVM2 identifier in its second sector (sector 1). This lets you use a different sector near the start of the disk (between 0 and 3 inclusive – see LABEL_SCAN_SECTORS in the source). Use with care.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

--[pv]metadatacopies 0|1|2

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1,

one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--repair

Repair headers and metadata on a PV.

--repairtype *pv_header|metadata|label_header*

Repair headers and metadata on a PV. See command description.

--settings *String*

Specifies command specific settings in "Key = Value" form. Combine multiple settings in quotes, or repeat the settings option for each.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

If the partition table is corrupted or lost on /dev/sda, and you suspect there was an LVM partition at approximately 100 MiB, then this area of the disk can be scanned using the --labelsector parameter with a value of 204800 ($100 * 1024 * 1024 / 512 = 204800$).

pvck --labelsector 204800 /dev/sda

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisk(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvcreate – Initialize physical volume(s) for use by LVM

SYNOPSIS

```
pvcreate position_args
      [ option_args ]
```

DESCRIPTION

pvcreate initializes a Physical Volume (PV) on a device so the device is recognized as belonging to LVM. This allows the PV to be used in a Volume Group (VG). An LVM disk label is written to the device, and LVM metadata areas are initialized. A PV can be placed on a whole device or partition.

Use **vgcreate(8)** to create a new VG on the PV, or **vgextend(8)** to add the PV to an existing VG. Use **pvr e-move(8)** to remove the LVM disk label from the device.

The force option will create a PV without confirmation. Repeating the force option (**-ff**) will forcibly create a PV, overriding checks that normally prevent it, e.g. if the PV is already in a VG.

Metadata location, size, and alignment

The LVM disk label begins 512 bytes from the start of the device, and is 512 bytes in size.

The LVM metadata area begins at an offset (from the start of the device) equal to the page size of the machine creating the PV (often 4 KiB.) The metadata area contains a 512 byte header and a multi-KiB circular buffer that holds text copies of the VG metadata.

With default settings, the first physical extent (PE), which contains LV data, is 1 MiB from the start of the device. This location is controlled by **default_data_alignment** in lvm.conf, which is set to 1 (MiB) by default. The pe_start will be a multiple of this many MiB. This location can be checked with:

```
pvs -o pe_start PV
```

The size of the LVM metadata area is the space between the the start of the metadata area and the first PE. When metadata begins at 4 KiB and the first PE is at 1024 KiB, the metadata area size is 1020 KiB. This can be checked with:

```
pvs -o mda_size PV
```

The mda_size cannot be increased after **pvcreate**, so if larger metadata is needed, it must be set during **pvcreate**. Two copies of the VG metadata must always fit within the metadata area, so the maximum VG metadata size is around half the mda_size. This can be checked with:

```
vgs -o mda_free VG
```

A larger metadata area can be set with **--metadatasize**. The resulting mda_size may be larger than specified due to **default_data_alignment** placing pe_start on a MiB boundary, and the fact that the metadata area extends to the first PE. With metadata starting at 4 KiB and **default_data_alignment** 1 (MiB), setting **--metadatasize** 2048k results in pe_start of 3 MiB and mda_size of 3068 KiB. Alternatively, **--metadatasize** 2044k results in pe_start at 2 MiB and mda_size of 2044 KiB.

The alignment of pe_start described above may be automatically overriden based on md device properties or device i/o properties reported in sysfs. These automatic adjustments can be enabled/disabled using lvm.conf settings **md_chunk_alignment** and **data_alignment_offset_detection**.

To use a different pe_start alignment, use the **--dataalignment** option. The **--metadatasize** option would also typically be used in this case because the metadata area size also determines the location of pe_start. When using these two options together, pe_start is calculated as: metadata area start (page size), plus the specified **--metadatasize**, rounded up to the next multiple of **--dataalignment**. With metadata starting at 4 KiB, **--metadatasize** 2048k, and **--dataalignment** 128k, pe_start is 2176 KiB and mda_size is 2172 KiB.

The pe_start of 2176 KiB is the nearest even multiple of 128 KiB that provides at least 2048 KiB of metadata space. Always check the resulting alignment and metadata size when using these options.

To shift an aligned pe_start value, use the --dataalignmentoffset option. The pe_start alignment is calculated as described above, and then the value specified with --dataalignmentoffset is added to produce the final pe_start value.

USAGE

```
pvcreate PV ...
[ -f|--force ]
[ -M|--metadatatype lvm2 ]
[ -u|--uuid String ]
[ -Z|--zero y|n ]
[ --dataalignment Size[k|UNIT] ]
[ --dataalignmentoffset Size[k|UNIT] ]
[ --bootloaderareasize Size[m|UNIT] ]
[ --labelsector Number ]
[ --[pv]metadatacopies 0|1|2 ]
[ --metadatasize Size[m|UNIT] ]
[ --metadataignore y|n ]
[ --norestorefile ]
[ --setphysicalvolumesize Size[m|UNIT] ]
[ --reportformat basic|json ]
[ --restorefile String ]
[ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--bootloaderareasize Size[m|UNIT]

Reserve space for the bootloader between the LVM metadata area and the first PE. The bootloader area is reserved for bootloaders to embed their own data or metadata; LVM will not use it. The bootloader area begins where the first PE would otherwise be located. The first PE is moved out by the size of the bootloader area, and then moved out further if necessary to match the data alignment. The start of the bootloader area is always aligned, see also --dataalignment and --dataalignmentoffset. The bootloader area may be larger than requested due to the alignment, but it's never less than the requested size. To see the bootloader area start and size of an existing PV use pvs -o +pv_ba_start,pv_ba_size.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--dataalignment *Size[k|UNIT]*

Align the start of a PV data area with a multiple of this number. To see the location of the first Physical Extent (PE) of an existing PV, use pvs -o +pe_start. In addition, it may be shifted by an alignment offset, see --dataalignmentoffset. Also specify an appropriate PE size when creating a VG.

--dataalignmentoffset *Size[k|UNIT]*

Shift the start of the PV data area by this additional offset.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--labelsector *Number*

By default the PV is labelled with an LVM2 identifier in its second sector (sector 1). This lets you use a different sector near the start of the disk (between 0 and 3 inclusive – see LA-BEL_SCAN_SECTORS in the source). Use with care.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--metadataignore *y|n*

Specifies the metadataignore property of a PV. If yes, metadata areas on the PV are ignored, and lvm will not store metadata in the metadata areas of the PV. If no, lvm will store metadata on the PV.

--metadatasize *Size[m|UNIT]*

The approximate amount of space used for each VG metadata area. The size may be rounded.

-M|--metadatatype *lvm2*

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format. **lvm1** (or just **1**) is no longer used.

--nolocking

Disable locking.

--norestorefile

In conjunction with --uuid, this allows a uuid to be specified without also requiring that a backup of the metadata be provided.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

--[pv]metadatacopies *0|1|2*

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1, one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG

metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--restorefile String

In conjunction with --uuid, this reads the file (produced by vgcfgbackup), extracts the location and size of the data on the PV, and ensures that the metadata produced by the program is consistent with the contents of the file, i.e. the physical extents will be in the same place and not be overwritten by new metadata. This provides a mechanism to upgrade the metadata format or to add/remove metadata areas. Use with care.

--setphysicalvolumesize Size[m|UNIT]

Overrides the automatically detected size of the PV. Use with care, or prior to reducing the physical size of the device.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-u|--uuid String

Specify a UUID for the device. Without this option, a random UUID is generated. This option is needed before restoring a backup of LVM metadata onto a replacement device; see **vgcfgrestore(8)**. As such, use of --restorefile is compulsory unless the --norestorefile is used. All PVs must have unique UUIDs, and LVM will prevent certain operations if multiple devices are seen with the same UUID. See **vgimportclone(8)** for more information.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

-Z|--zero y|n

Controls if the first 4 sectors (2048 bytes) of the device are wiped. The default is to wipe these sectors unless either or both of --restorefile or --uuid are specified.

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Initialize a partition and a full device.

```
pvcreate /dev/sdc4 /dev/sde
```

If a device is a 4KiB sector drive that compensates for windows partitioning (sector 7 is the lowest aligned logical block, the 4KiB sectors start at LBA -1, and consequently sector 63 is aligned on a 4KiB boundary) manually account for this when initializing for use by LVM.

```
pvcreate --dataalignmentoffset 7s /dev/sdb
```

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvdisplay – Display various attributes of physical volume(s)

SYNOPSIS

```
pvdisplay
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

pvdisplay shows the attributes of PVs, like size, physical extent size, space used for the VG descriptor area, etc.

pvs(8) is a preferred alternative that shows the same information and more, using a more compact and configurable output format.

USAGE

```
pvdisplay
  [ -a|--all ]
  [ -c|--colon ]
  [ -C|--columns ]
  [ -m|--maps ]
  [ -o|--options String ]
  [ -S|--select String ]
  [ -s|--short ]
  [ -O|--sort String ]
  [ --aligned ]
  [ --binary ]
  [ --configreport log|vg|lv|pv|pvseg|seg ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --logonly ]
  [ --noheadings ]
  [ --nosuffix ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ --separator String ]
  [ --shared ]
  [ --unbuffered ]
  [ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
  [ COMMON_OPTIONS ]
  [ PV|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
```

[--version]

OPTIONS

--aligned

Use with --separator to align the output columns

-a|--all

Show information about devices that have not been initialized by LVM, i.e. they are not PVs.

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

-c|--colon

Generate colon separated output for easier parsing in scripts or programs. Also see **vgs(8)** which provides considerably more control over the output.

-C|--columns

Display output in columns, the equivalent of **vgs(8)**. Options listed are the same as options given in **vgs(8)**.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport log|vg|lv|pv|pvseg|seg

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

-m|--maps

Display the mapping of physical extents to LVs and logical extents.

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with --units (except h and H) if processing the output.

-o|--options String

Comma-separated, ordered list of fields to display in columns. String arg syntax is:

[+|-#]Field1[,Field2 ...] The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the -o option: -o+field1,field2 -o-field3,field4 -o#field5. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **lvmreport(7)** for more information about reporting.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator String

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when lvmlockd is not being used on the host. See **lvmlockd(8)** for more information about shared VGs.

-s|--short

Only display the size of the given PVs.

-O|--sort String

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any column with - for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes

has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-read-able, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes.

Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g.
--units 3M.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

Tag

Tag name. See lvm(8) for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See lvm(8) for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvmove – Move extents from one physical volume to another

SYNOPSIS

```
pvmove position_args
      [ option_args ]
      [ position_args ]
```

DESCRIPTION

pvmove moves the allocated physical extents (PEs) on a source PV to one or more destination PVs. You can optionally specify a source LV in which case only extents used by that LV will be moved to free (or specified) extents on the destination PV. If no destination PV is specified, the normal allocation rules for the VG are used.

If **pvmove** is interrupted for any reason (e.g. the machine crashes) then run **pvmove** again without any PV arguments to restart any operations that were in progress from the last checkpoint. Alternatively, use the abort option at any time to abort the operation. The resulting location of LVs after an abort depends on whether the atomic option was used.

More than one **pvmove** can run concurrently if they are moving data from different source PVs, but additional **pvmoves** will ignore any LVs already in the process of being changed, so some data might not get moved.

USAGE

Move PV extents.

```
pvmove PV
      [ -A|--autobackup y|n ]
      [ -n|--name LV ]
      [ --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit ]
      [ --atomic ]
      [ --noudevsync ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
      [ PV ... ]
```

Continue or abort existing **pvmove** operations.

```
pvmove
      [ COMMON_OPTIONS ]
```

Common options for command:

```
[ -b|--background ]
[ -i|--interval Number ]
[ --abort ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
```

```
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--abort**

Abort any pvmove operations in progress. If a pvmove was started with the --atomic option, then all LVs will remain on the source PV. Otherwise, segments that have been moved will remain on the destination PV, while unmoved segments will remain on the source PV.

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

--atomic

Makes a pvmove operation atomic, ensuring that all affected LVs are moved to the destination PV, or none are if the operation is aborted.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

-b|--background

If the operation requires polling, this option causes the command to return before the operation is complete, and polling is done in the background.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

-i|--interval Number

Report progress at regular intervals.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-n|--name *String*

Move only the extents belonging to the named LV.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

pvmove works as follows:

1. A temporary 'pvmove' LV is created to store details of all the data movements required.
2. Every LV in the VG is searched for contiguous data that need moving according to the command line arguments. For each piece of data found, a new segment is added to the end of the pvmove LV. This segment takes the form of a temporary mirror to copy the data from the original location to a newly allocated location. The original LV is updated to use the new temporary mirror segment in the pvmove LV instead of accessing the data directly.
3. The VG metadata is updated on disk.
4. The first segment of the pvmove LV is activated and starts to mirror the first part of the data. Only one segment is mirrored at once as this is usually more efficient.
5. A daemon repeatedly checks progress at the specified time interval. When it detects that the first temporary mirror is in sync, it breaks that mirror so that only the new location for that data gets used and writes a checkpoint into the VG metadata on disk. Then it activates the mirror for the next segment of the pvmove LV.
6. When there are no more segments left to be mirrored, the temporary LV is removed and the VG metadata is updated so that the LVs reflect the new data locations.

Note that this new process cannot support the original LVM1 type of on-disk metadata. Metadata can be converted using **vgconvert(8)**.

If the **--atomic** option is used, a slightly different approach is used for the move. Again, a temporary 'pvmove' LV is created to store the details of all the data movements required. This temporary LV contains all the segments of the various LVs that need to be moved. However, in this case, an identical LV is allocated that contains the same number of segments and a mirror is created to copy the contents from the first temporary LV to the second. After a complete copy is made, the temporary LVs are removed, leaving behind the segments on the destination PV. If an abort is issued during the move, all LVs being moved will remain on the source PV.

EXAMPLES

Move all physical extents that are used by simple LVs on the specified PV to free physical extents elsewhere in the VG.

pvmove /dev/sdb1

Use a specific destination PV when moving physical extents.

pvmove /dev/sdb1 /dev/sdc1

Move extents belonging to a single LV.

pvmove -n lvol1 /dev/sdb1 /dev/sdc1

Rather than moving the contents of an entire device, it is possible to move a range of physical extents, for example numbers 1000 to 1999 inclusive on the specified PV.

pvmove /dev/sdb1:1000-1999

A range of physical extents to move can be specified as start+length. For example, starting from PE 1000.

(Counting starts from 0, so this refers to the 1001st to the 2000th PE inclusive.)

pvmmove /dev/sdb1:1000+1000

Move a range of physical extents to a specific PV (which must have sufficient free extents).

pvmmove /dev/sdb1:1000-1999 /dev/sdc1

Move a range of physical extents to specific new extents on a new PV.

pvmmove /dev/sdb1:1000-1999 /dev/sdc1:0-999

If the source and destination are on the same disk, the **anywhere** allocation policy is needed.

pvmmove --alloc anywhere /dev/sdb1:1000-1999 /dev/sdb1:0-999

The part of a specific LV present within in a range of physical extents can also be picked out and moved.

pvmmove -n lvol1 /dev/sdb1:1000-1999 /dev/sdc1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvremove – Remove LVM label(s) from physical volume(s)

SYNOPSIS

```
pvremove position_args
      [ option_args ]
```

DESCRIPTION

pvremove wipes the label on a device so that LVM will no longer recognise it as a PV.

A PV cannot be removed from a VG while it is used by an active LV.

Repeat the force option (**-ff**) to forcibly remove a PV belonging to an existing VG. Normally, **vgreduce(8)** should be used instead.

USAGE

```
pvremove PV ...
      [ -f|--force ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--commandprofile** *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*
 Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp
 Display long help text.

--nolocking
 Disable locking.

--profile *String*
 An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...
 Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json
 Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test
 Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...
 Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version
 Display version information.

-y|--yes
 Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

PV
String
 Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String
 See the option description for information about the string content.

Size[UNIT]
 Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgecfgbackup(8) vgecfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvresize – Resize physical volume(s)

SYNOPSIS

```
pvresize position_args
      [ option_args ]
```

DESCRIPTION

pvresize resizes a PV. The PV may already be in a VG and may have active LVs allocated on it.

USAGE

```
pvresize PV ...
      [ --setphysicalvolumesize Size[m|UNIT] ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ --d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded **y|n**

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--setphysicalvolumesize *Size[m|UNIT]*

Overrides the automatically detected size of the PV. Use with care, or prior to reducing the physical size of the device.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

pvresize will refuse to shrink a PV if it has allocated extents beyond the new end.

EXAMPLES

Expand a PV after enlarging the partition.

```
pvresize /dev/sda1
```

Shrink a PV prior to shrinking the partition (ensure that the PV size is appropriate for the intended new partition size).

```
pvresize --setphysicalvolumesize 40G /dev/sda1
```

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvs – Display information about physical volumes

SYNOPSIS

```
pvs
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

pvs produces formatted output about PVs.

USAGE

```
pvs
  [ -a|--all ]
  [ -o|--options String ]
  [ -S|--select String ]
  [ -O|--sort String ]
  [ --segments ]
  [ --aligned ]
  [ --binary ]
  [ --configreport log|vg|lv|pv|pvseg|seg ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --logonly ]
  [ --nameprefixes ]
  [ --noheadings ]
  [ --nosuffix ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ --rows ]
  [ --separator String ]
  [ --shared ]
  [ --unbuffered ]
  [ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
  [ --unquoted ]
  [ COMMON_OPTIONS ]
  [ PV|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--aligned**

Use with **--separator** to align the output columns

-a|--all

Show information about devices that have not been initialized by LVM, i.e. they are not PVs.

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport log|vg|lv|pv|pvseg|seg

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

--nameprefixes

Add an "LVM2_" prefix plus the field name to the output. Useful with --noheadings to produce a list of field=value pairs that can be used to set environment variables (for example, in udev rules).

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with --units (except h and H) if processing the output.

-o|--options *String*

Comma-separated, ordered list of fields to display in columns. String arg syntax is:
`[+|-#]Field1[,Field2 ...]` The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the -o option: `-o+field1,field2 -o-`

field3,field4 -o#field5. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **lvmreport(7)** for more information about reporting.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--rows

Output columns as rows.

--segments

Produces one line of output for each contiguous allocation of space on each PV, showing the start (pvseg_start) and length (pvseg_size) in units of physical extents.

-S|--select *String*

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator *String*

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when lvmlockd is not being used on the host. See **lvmlockd(8)** for more information about shared VGs.

-O|--sort *String*

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any column with - for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units **r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E**

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-readable, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes.

Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g.
—units 3M.

--unquoted

When used with --nameprefixes, output values in the field=value pairs are not quoted.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE–PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control —units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

The pv_attr bits are:

- 1 (d)uplicate, (a)llocatable, (u)sed
- 2 e(x)ported
- 3 (m)issing

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pvscan – List all physical volumes

SYNOPSIS

```
pvscan option_args
      [ option_args ]
      [ position_args ]
```

DESCRIPTION

When called without the **--cache** option, **pvscan** lists PVs on the system, like **pvs(8)** or **pvdisplay(8)**.

When the **--cache** and **-aay** options are used, **pvscan** records which PVs are available on the system, and activates LVs in completed VGs. A VG is complete when **pvscan** sees that the final PV in the VG has appeared. This is used by event-based system startup (systemd, udev) to activate LVs.

The four main variations of this are:

pvscan --cache device

If *device* is present, lvm adds a record that the PV on *device* is online. If *device* is not present, lvm removes the online record for the PV. In most cases, the **pvscan** will only read the named devices.

pvscan --cache -aay device...

This begins by performing the same steps as above. Afterward, if the VG for the specified PV is complete, then **pvscan** will activate LVs in the VG (the same as **vgchange -aay vgname** would do.)

pvscan --cache

This first clears all existing PV online records, then scans all devices on the system, adding PV online records for any PVs that are found.

pvscan --cache -aay

This begins by performing the same steps as **pvscan --cache**. Afterward, it activates LVs in any complete VGs.

To prevent devices from being scanned by **pvscan --cache**, add them to **lvm.conf(5) devices/global_filter**. For more information, see:

lvmconfig --withcomments devices/global_filter

Auto-activation of VGs or LVs can be enabled/disabled using:

lvm.conf(5) activation/auto_activation_volume_list

For more information, see:

lvmconfig --withcomments activation/auto_activation_volume_list

To disable auto-activation, explicitly set this list to an empty list, i.e. **auto_activation_volume_list = []**.

When this setting is undefined (e.g. commented), then all LVs are auto-activated.

USAGE

Display PV information.

```
pvscan
      [ -e|--exported ]
```

```
[ -n|--novolumegroup ]
[ -s|--short ]
[ -u|--uuid ]
[ COMMON_OPTIONS ]
```

Autoactivate a VG when all PVs are online.

```
pvscan --cache
[ -a|--activate ay ]
[ -j|--major Number ]
[ --minor Number ]
[ --noudevsync ]
[ COMMON_OPTIONS ]
[ String|PV ... ]
```

Common options for command:

```
[ --ignorelockingfailure ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-a|--activate y|n|ay

Auto-activate LVs in a VG when the PVs scanned have completed the VG. (Only **ay** is applicable.)

--cache

Scan one or more devices and record that they are online.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-e|--exported

Only show PVs belonging to exported VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-j|--major Number

The major number of a device.

--minor Number

The minor number of a device.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

-n|--novolumegroup

Only show PVs not belonging to any VG.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-s|--short

Short listing format.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-u|--uuid

Show UUIDs in addition to device names.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **--qq**.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **Ivm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

pwck – verify integrity of password files

SYNOPSIS

pwck [options] [*passwd* [*shadow*]]

DESCRIPTION

The **pwck** command verifies the integrity of the users and authentication information. It checks that all entries in /etc/passwd and /etc/shadow have the proper format and contain valid data. The user is prompted to delete entries that are improperly formatted or which have other uncorrectable errors.

Checks are made to verify that each entry has:

- the correct number of fields
- a unique and valid user name
- a valid user and group identifier
- a valid primary group
- a valid home directory
- a valid login shell

shadow checks are enabled when a second file parameter is specified or when /etc/shadow exists on the system.

These checks are the following:

- every passwd entry has a matching shadow entry, and every shadow entry has a matching passwd entry
- passwords are specified in the shadowed file
- shadow entries have the correct number of fields
- shadow entries are unique in shadow
- the last password changes are not in the future

The checks for correct number of fields and unique user name are fatal. If the entry has the wrong number of fields, the user will be prompted to delete the entire line. If the user does not answer affirmatively, all further checks are bypassed. An entry with a duplicated user name is prompted for deletion, but the remaining checks will still be made. All other errors are warning and the user is encouraged to run the **usermod** command to correct the error.

The commands which operate on the /etc/passwd file are not able to alter corrupted or duplicated entries. **pwck** should be used in those circumstances to remove the offending entry.

OPTIONS

The **-r** and **-s** options cannot be combined.

The options which apply to the **pwck** command are:

--badname

Allow names that do not conform to standards.

-h, --help

Display help message and exit.

-q, --quiet

Report errors only. The warnings which do not require any action from the user won't be displayed.

-r, --read-only

Execute the **pwck** command in read-only mode.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR*

directory.

-s, --sort

Sort entries in /etc/passwd and /etc/shadow by UID.

By default, **pwck** operates on the files /etc/passwd and /etc/shadow. The user may select alternate files with the *passwd* and *shadow* parameters.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

PASS_MAX_DAYS (number)

The maximum number of days a password may be used. If the password is older than this, a password change will be forced. If not specified, -1 will be assumed (which disables the restriction).

PASS_MIN_DAYS (number)

The minimum number of days allowed between password changes. Any password changes attempted sooner than this will be rejected. If not specified, -1 will be assumed (which disables the restriction).

PASS_WARN_AGE (number)

The number of days warning given before a password expires. A zero means warning is given only upon the day of expiration, a negative value means no warning is given. If not specified, no warning will be provided.

FILES

/etc/group

Group account information.

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

EXIT VALUES

The **pwck** command exits with the following values:

0

success

1

invalid command syntax

2

one or more bad password entries

3

can't open password files

4

can't lock password files

5

can't update password files

6

can't sort password files

SEE ALSO

group(5), grpck(8), passwd(5), shadow(5), usermod(8).

NAME

rarp – manipulate the system RARP table

SYNOPSIS

```
rarp [-V] [--version] [-h] [--help]
rarp -a
rarp [-v] -d hostname ...
rarp [-v] [-t type] -s hostname hw_addr
```

NOTE

This program is obsolete. From version 2.3, the Linux kernel no longer contains RARP support. For a replacement RARP daemon, see <ftp://ftp.dementia.org/pub/net-tools>

DESCRIPTION

Rarp manipulates the kernel's RARP table in various ways. The primary options are clearing an address mapping entry and manually setting up one. For debugging purposes, the **rarp** program also allows a complete dump of the RARP table.

OPTIONS

- V** Display the version of RARP in use.
- v** Tell the user what is going on by being verbose.
- t type** When setting or reading the RARP table, this optional parameter tells **rarp** which class of entries it should check for. The default value of this parameter is **ether** (i.e. hardware code **0x01** for **IEEE 802.3 10Mbps Ethernet**). Other values might include network technologies such as **AX.25** (**ax25**) and **NET/ROM (netrom)**.
- a**
- list** Lists the entries in the RARP table.
- d hostname**
- delete hostname**
Remove all RARP entries for the specified host.
- s hostname hw_addr**
- set hostname hw_addr**
Create a RARP address mapping entry for host **hostname** with hardware address set to **hw_addr**.
The format of the hardware address is dependent on the hardware class, but for most classes one can assume that the usual presentation can be used. For the Ethernet class, this is 6 bytes in hexadecimal, separated by colons.

WARNING

Some systems (notably older Suns) assume that the host replying to a RARP query can also provide other remote boot services. Therefore never gratuitously add rarp entries unless you wish to meet the wrath of the network administrator.

FILES

/proc/net/rarp,

SEE ALSO

arp(8), route(8), ifconfig(8), netstat(8)

AUTHORS

Ross D. Martin, <martin@trcsun3.eas.asu.edu>
Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
Phil Blundell, <Philip.Blundell@pobox.com>

NAME

systemd-rc-local-generator, rc-local.service – Compatibility generator and service to start /etc/rc.local during boot

SYNOPSIS

```
/lib/systemd/system-generators/systemd-rc-local-generator  
rc-local.service
```

DESCRIPTION

systemd-rc-local-generator is a generator that checks whether /etc/rc.local exists and is executable, and if it is, pulls the rc-local.service unit into the boot process. This unit is responsible for running this script during late boot. The script is run after network.target, but in parallel with most other regular system services.

Note that rc-local.service runs with slightly different semantics than the original System V version, which was executed "last" in the boot process, which is a concept that does not translate to systemd.

Also note that rc-local.service is ordered after network.target, which does not mean that the network is functional, see **systemd.special(7)**. If the script requires a configured network connection, it may be desirable to pull in and order it after network-online.target with a drop-in:

```
# /etc/systemd/system/rc-local.service.d/network.conf  
[Unit]  
Wants=network-online.target  
After=network-online.target
```

Support for /etc/rc.local is provided for compatibility with specific System V systems only. However, it is strongly recommended to avoid making use of this script today, and instead provide proper unit files with appropriate dependencies for any scripts to run during the boot process. Note that the path to the script is set at compile time and varies between distributions.

systemd-rc-local-generator implements **systemd.generator(7)**.

SEE ALSO

systemd(1), **systemctl(1)**

NAME

`regcomp`, `regexec`, `reerror`, `regfree` – POSIX regex functions

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <regex.h>

int regcomp(regex_t *restrict preg, const char *restrict regex,
            int cflags);
int regexec(const regex_t *restrict preg, const char *restrict string,
            size_t nmatch, regmatch_t pmatch[restrict .nmatch],
            int eflags);
size_t reerror(int errcode, const regex_t *restrict preg,
               char errbuf[restrict .errbuf_size], size_t errbuf_size);
void regfree(regex_t *preg);
```

DESCRIPTION**POSIX regex compiling**

`regcomp()` is used to compile a regular expression into a form that is suitable for subsequent `regexec()` searches.

`regcomp()` is supplied with *preg*, a pointer to a pattern buffer storage area; *regex*, a pointer to the null-terminated string and *cflags*, flags used to determine the type of compilation.

All regular expression searching must be done via a compiled pattern buffer, thus `regexec()` must always be supplied with the address of a `regcomp()`-initialized pattern buffer.

cflags is the bitwise-**or** of zero or more of the following:

REG_EXTENDED

Use **POSIX** Extended Regular Expression syntax when interpreting *regex*. If not set, **POSIX** Basic Regular Expression syntax is used.

REG_ICASE

Do not differentiate case. Subsequent `regexec()` searches using this pattern buffer will be case insensitive.

REG_NOSUB

Do not report position of matches. The *nmatch* and *pmatch* arguments to `regexec()` are ignored if the pattern buffer supplied was compiled with this flag set.

REG_NEWLINE

Match-any-character operators don't match a newline.

A nonmatching list ([^...]) not containing a newline does not match a newline.

Match-beginning-of-line operator (^) matches the empty string immediately after a newline, regardless of whether *eflags*, the execution flags of `regexec()`, contains **REG_NOTBOL**.

Match-end-of-line operator (\$) matches the empty string immediately before a newline, regardless of whether *eflags* contains **REG_NOTEOL**.

POSIX regex matching

`regexec()` is used to match a null-terminated string against the precompiled pattern buffer, *preg*. *nmatch* and *pmatch* are used to provide information regarding the location of any matches. *eflags* is the bitwise-**or** of zero or more of the following flags:

REG_NOTBOL

The match-beginning-of-line operator always fails to match (but see the compilation flag **REG_NEWLINE** above). This flag may be used when different portions of a string are passed to `regexec()` and the beginning of the string should not be interpreted as the beginning of the line.

REG_NOTEOL

The match-end-of-line operator always fails to match (but see the compilation flag **REG_NEWLINE** above).

REG_STARTEND

Use *pmatch[0]* on the input string, starting at byte *pmatch[0].rm_so* and ending before byte *pmatch[0].rm_eo*. This allows matching embedded NUL bytes and avoids a **strlen(3)** on large strings. It does not use *nmatch* on input, and does not change **REG_NOTBOL** or **REG_NEWLINE** processing. This flag is a BSD extension, not present in POSIX.

Byte offsets

Unless **REG_NOSUB** was set for the compilation of the pattern buffer, it is possible to obtain match addressing information. *pmatch* must be dimensioned to have at least *nmatch* elements. These are filled in by **regexec()** with substring match addresses. The offsets of the subexpression starting at the *i*th open parenthesis are stored in *pmatch[i]*. The entire regular expression's match addresses are stored in *pmatch[0]*. (Note that to return the offsets of *N* subexpression matches, *nmatch* must be at least *N+1*.) Any unused structure elements will contain the value -1.

The *regmatch_t* structure which is the type of *pmatch* is defined in *<regex.h>*.

```
typedef struct {
    regoff_t rm_so;
    regoff_t rm_eo;
} regmatch_t;
```

Each *rm_so* element that is not -1 indicates the start offset of the next largest substring match within the string. The relative *rm_eo* element indicates the end offset of the match, which is the offset of the first character after the matching text.

POSIX error reporting

reerror() is used to turn the error codes that can be returned by both **regcomp()** and **regexec()** into error message strings.

reerror() is passed the error code, *errcode*, the pattern buffer, *preg*, a pointer to a character string buffer, *errbuf*, and the size of the string buffer, *errbuf_size*. It returns the size of the *errbuf* required to contain the null-terminated error message string. If both *errbuf* and *errbuf_size* are nonzero, *errbuf* is filled in with the first *errbuf_size* - 1 characters of the error message and a terminating null byte ('\0').

POSIX pattern buffer freeing

Supplying **regfree()** with a precompiled pattern buffer, *preg*, will free the memory allocated to the pattern buffer by the compiling process, **regcomp()**.

RETURN VALUE

regcomp() returns zero for a successful compilation or an error code for failure.

regexec() returns zero for a successful match or **REG_NOMATCH** for failure.

ERRORS

The following errors can be returned by **regcomp()**:

REG_BADBR

Invalid use of back reference operator.

REG_BADPAT

Invalid use of pattern operators such as group or list.

REG_BADRPT

Invalid use of repetition operators such as using '*' as the first character.

REG_EBRACE

Un-matched brace interval operators.

REG_EBRACK

Un-matched bracket list operators.

REG_ECOLLATE

Invalid collating element.

REG_ECTYPE

Unknown character class name.

REG_EEND

Nonspecific error. This is not defined by POSIX.2.

REG_EESCAPE

Trailing backslash.

REG_EPAREN

Un-matched parenthesis group operators.

REG_ERANGE

Invalid use of the range operator; for example, the ending point of the range occurs prior to the starting point.

REG_ESIZE

Compiled regular expression requires a pattern buffer larger than 64 kB. This is not defined by POSIX.2.

REG_ESPACE

The regex routines ran out of memory.

REG_ESUBREG

Invalid back reference to a subexpression.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
<code>regcomp()</code> , <code>regexec()</code>	Thread safety	MT-Safe locale
<code>regerror()</code>	Thread safety	MT-Safe env
<code>regfree()</code>	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008.

EXAMPLES

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>

#define ARRAY_SIZE(arr) (sizeof((arr)) / sizeof((arr)[0]))

static const char *const str =
    "1) John Driverhacker;\n2) John Doe;\n3) John Foo;\n";
static const char *const re = "John.*o";

int main(void)
{
    static const char *s = str;
    regex_t      regex;
    regmatch_t   pmatch[1];
    regoff_t     off, len;
```

```
if (regcomp(&regex, re, REG_NEWLINE))
    exit(EXIT_FAILURE);

printf("String = \"%s\"\n", str);
printf("Matches:\n");

for (unsigned int i = 0; ; i++) {
    if (regexec(&regex, s, ARRAY_SIZE(pmatch), pmatch, 0))
        break;

    off = pmatch[0].rm_so + (s - str);
    len = pmatch[0].rm_eo - pmatch[0].rm_so;
    printf("#%zu:\n", i);
    printf("offset = %jd; length = %jd\n", (intmax_t) off,
           (intmax_t) len);
    printf("substring = \"%.*s\"\n", len, s + pmatch[0].rm_so);

    s += pmatch[0].rm_eo;
}

exit(EXIT_SUCCESS);
}
```

SEE ALSO

grep(1), regex(7)

The glibc manual section, *Regular Expressions*

NAME

regex – POSIX.2 regular expressions

DESCRIPTION

Regular expressions ("RE"s), as defined in POSIX.2, come in two forms: modern REs (roughly those of *egrep*; POSIX.2 calls these "extended" REs) and obsolete REs (roughly those of *ed*(1); POSIX.2 "basic" REs). Obsolete REs mostly exist for backward compatibility in some old programs; they will be discussed at the end. POSIX.2 leaves some aspects of RE syntax and semantics open; "†" marks decisions on these aspects that may not be fully portable to other POSIX.2 implementations.

A (modern) RE is one† or more nonempty† *branches*, separated by '|'. It matches anything that matches one of the branches.

A branch is one† or more *pieces*, concatenated. It matches a match for the first, followed by a match for the second, and so on.

A piece is an *atom* possibly followed by a single† '*', '+', '?', or *bound*. An atom followed by '*' matches a sequence of 0 or more matches of the atom. An atom followed by '+' matches a sequence of 1 or more matches of the atom. An atom followed by '?' matches a sequence of 0 or 1 matches of the atom.

A *bound* is '{' followed by an unsigned decimal integer, possibly followed by ',' possibly followed by another unsigned decimal integer, always followed by '}'. The integers must lie between 0 and **RE_DUP_MAX** (255†) inclusive, and if there are two of them, the first may not exceed the second. An atom followed by a bound containing one integer *i* and no comma matches a sequence of exactly *i* matches of the atom. An atom followed by a bound containing one integer *i* and a comma matches a sequence of *i* or more matches of the atom. An atom followed by a bound containing two integers *i* and *j* matches a sequence of *i* through *j* (inclusive) matches of the atom.

An atom is a regular expression enclosed in "()" (matching a match for the regular expression), an empty set of "()" (matching the null string)†, a *bracket expression* (see below), '.' (matching any single character), '^' (matching the null string at the beginning of a line), '\$' (matching the null string at the end of a line), a '\' followed by one of the characters ".[\$()/*+?/]" (matching that character taken as an ordinary character), a '\' followed by any other character† (matching that character taken as an ordinary character, as if the '\' had not been present†), or a single character with no other significance (matching that character). A '{' followed by a character other than a digit is an ordinary character, not the beginning of a bound†. It is illegal to end an RE with '\'.

A *bracket expression* is a list of characters enclosed in "[]". It normally matches any single character from the list (but see below). If the list begins with '^', it matches any single character (but see below) *not* from the rest of the list. If two characters in the list are separated by '-', this is shorthand for the full *range* of characters between those two (inclusive) in the collating sequence, for example, "[0-9]" in ASCII matches any decimal digit. It is illegal† for two ranges to share an endpoint, for example, "a-c-e". Ranges are very collating-sequence-dependent, and portable programs should avoid relying on them.

To include a literal ']' in the list, make it the first character (following a possible '^'). To include a literal '-', make it the first or last character, or the second endpoint of a range. To use a literal '-' as the first endpoint of a range, enclose it in "[." and ".]" to make it a collating element (see below). With the exception of these and some combinations using '[' (see next paragraphs), all other special characters, including '\', lose their special significance within a bracket expression.

Within a bracket expression, a collating element (a character, a multicharacter sequence that collates as if it were a single character, or a collating-sequence name for either) enclosed in "[." and ".]" stands for the sequence of characters of that collating element. The sequence is a single element of the bracket expression's list. A bracket expression containing a multicharacter collating element can thus match more than one character, for example, if the collating sequence includes a "ch" collating element, then the RE "[.ch.]*c" matches the first five characters of "chchcc".

Within a bracket expression, a collating element enclosed in "[=" and "=]" is an equivalence class, standing for the sequences of characters of all collating elements equivalent to that one, including itself. (If there are no other equivalent collating elements, the treatment is as if the enclosing delimiters were "[." and ".]" .)

For example, if o and ô are the members of an equivalence class, then "[=o=]", "[=ô=]", and "[oô]" are all synonymous. An equivalence class may not† be an endpoint of a range.

Within a bracket expression, the name of a *character class* enclosed in ":" and ":" stands for the list of all characters belonging to that class. Standard character class names are:

alnum	digit	punct
alpha	graph	space
blank	lower	upper
cntrl	print	xdigit

These stand for the character classes defined in **wctype(3)**. A locale may provide others. A character class may not be used as an endpoint of a range.

In the event that an RE could match more than one substring of a given string, the RE matches the one starting earliest in the string. If the RE could match more than one substring starting at that point, it matches the longest. Subexpressions also match the longest possible substrings, subject to the constraint that the whole match be as long as possible, with subexpressions starting earlier in the RE taking priority over ones starting later. Note that higher-level subexpressions thus take priority over their lower-level component subexpressions.

Match lengths are measured in characters, not collating elements. A null string is considered longer than no match at all. For example, "bb*" matches the three middle characters of "abbbc", "(wee/week)(knights/nights)" matches all ten characters of "weeknights", when "(.*).*?" is matched against "abc" the parenthesized subexpression matches all three characters, and when "(a*)*?" is matched against "bc" both the whole RE and the parenthesized subexpression match the null string.

If case-independent matching is specified, the effect is much as if all case distinctions had vanished from the alphabet. When an alphabetic that exists in multiple cases appears as an ordinary character outside a bracket expression, it is effectively transformed into a bracket expression containing both cases, for example, 'x' becomes "[xX]". When it appears inside a bracket expression, all case counterparts of it are added to the bracket expression, so that, for example, "[x]" becomes "[xX]" and "[^x]" becomes "[^xX]".

No particular limit is imposed on the length of REs†. Programs intended to be portable should not employ REs longer than 256 bytes, as an implementation can refuse to accept such REs and remain POSIX-compliant.

Obsolete ("basic") regular expressions differ in several respects. '|', '+', and '?' are ordinary characters and there is no equivalent for their functionality. The delimiters for bounds are "\/" and "\/", with '{' and '}' by themselves ordinary characters. The parentheses for nested subexpressions are "\(" and "\)", with '(' and ')' by themselves ordinary characters. '^' is an ordinary character except at the beginning of the RE or† the beginning of a parenthesized subexpression, '\$' is an ordinary character except at the end of the RE or† the end of a parenthesized subexpression, and '*' is an ordinary character if it appears at the beginning of the RE or the beginning of a parenthesized subexpression (after a possible leading '^').

Finally, there is one new type of atom, a *back reference*: '\' followed by a nonzero decimal digit d matches the same sequence of characters matched by the dth parenthesized subexpression (numbering subexpressions by the positions of their opening parentheses, left to right), so that, for example, "\([bc]\)\I" matches "bb" or "cc" but not "bc".

BUGS

Having two kinds of REs is a botch.

The current POSIX.2 spec says that ')' is an ordinary character in the absence of an unmatched '('; this was an unintentional result of a wording error, and change is likely. Avoid relying on it.

Back references are a dreadful botch, posing major problems for efficient implementations. They are also somewhat vaguely defined (does "a\(\(b\)\)*2\)*d" match "abbbd"?). Avoid using them.

POSIX.2's specification of case-independent matching is vague. The "one case implies all cases" definition given above is current consensus among implementors as to the right interpretation.

AUTHOR

This page was taken from Henry Spencer's regex package.

SEE ALSO

grep(1), regex(3)

POSIX.2, section 2.8 (Regular Expression Notation).

NAME

`system_data_types` – overview of system data types

DESCRIPTION

aiocb

Include: `<aio.h>`.

```
struct aiocb {
    int             aio_fildes;      /* File descriptor */
    off_t           aio_offset;     /* File offset */
    volatile void * aio_buf;        /* Location of buffer */
    size_t          aio_nbytes;     /* Length of transfer */
    int             aio_reqprio;    /* Request priority offset */
    struct sigevent aio_sigevent;  /* Signal number and value */
    int             aio_lio_opcode; /* Operation to be performed */
};
```

For further information about this structure, see [aio\(7\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [aio_cancel\(3\)](#), [aio_error\(3\)](#), [aio_fsync\(3\)](#), [aio_read\(3\)](#), [aio_return\(3\)](#), [aio_suspend\(3\)](#), [aio_write\(3\)](#), [lio_listio\(3\)](#)

clock_t

Include: `<time.h>` or `<sys/types.h>`. Alternatively, `<sys/time.h>`.

Used for system time in clock ticks or **CLOCKS_PER_SEC** (defined in `<time.h>`). According to POSIX, it shall be an integer type or a real-floating type.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [times\(2\)](#), [clock\(3\)](#)

clockid_t

Include: `<sys/types.h>`. Alternatively, `<time.h>`.

Used for clock ID type in the clock and timer functions. According to POSIX, it shall be defined as an arithmetic type.

Conforming to: POSIX.1-2001 and later.

See also: [clock_adjtime\(2\)](#), [clock_getres\(2\)](#), [clock_nanosleep\(2\)](#), [timer_create\(2\)](#), [clock_getcpuclockid\(3\)](#)

dev_t

Include: `<sys/types.h>`. Alternatively, `<sys/stat.h>`.

Used for device IDs. According to POSIX, it shall be an integer type. For further details of this type, see [makedev\(3\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [mknod\(2\)](#), [stat\(2\)](#)

div_t

Include: `<stdlib.h>`.

```
typedef struct {
    int quot; /* Quotient */
    int rem; /* Remainder */
} div_t;
```

It is the type of the value returned by the [div\(3\)](#) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [div\(3\)](#)

double_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *double*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *double_t* is *double*.
- 1 *double_t* is *double*.
- 2 *double_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *double_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *float_t* type in this page.

fd_set

Include: <sys/select.h>. Alternatively, <sys/time.h>.

A structure type that can represent a set of file descriptors. According to POSIX, the maximum number of file descriptors in an *fd_set* structure is the value of the macro **FD_SETSIZE**.

Conforming to: POSIX.1-2001 and later.

See also: **select**(2)

fenv_t

Include: <fenv.h>.

This type represents the entire floating-point environment, including control modes and status flags; for further details, see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

fexcept_t

Include: <fenv.h>.

This type represents the floating-point status flags collectively; for further details see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

FILE

Include: <stdio.h>. Alternatively, <wchar.h>.

An object type used for streams.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fclose**(3), **flockfile**(3), **fopen**(3), **fprintf**(3), **fread**(3), **fscanf**(3), **stdin**(3), **stdio**(3)

float_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *float*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *float_t* is *float*.
- 1 *float_t* is *double*.
- 2 *float_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *float_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *double_t* type in this page.

gid_t

Include: <sys/types.h>. Alternatively, <grp.h>, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold group IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: chown(2), getgid(2), getegid(2), getgroups(2), getresgid(2), getgrnam(2), credentials(7)

id_t

Include: <sys/types.h>. Alternatively, <sys/resource.h>.

A type used to hold a general identifier. According to POSIX, this shall be an integer type that can be used to contain a *pid_t*, *uid_t*, or *gid_t*.

Conforming to: POSIX.1-2001 and later.

See also: getpriority(2), waitid(2)

imaxdiv_t

Include: <inttypes.h>.

```
typedef struct {
    intmax_t      quot; /* Quotient */
    intmax_t      rem;   /* Remainder */
} imaxdiv_t;
```

It is the type of the value returned by the imaxdiv(3) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: imaxdiv(3)

intmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

A signed integer type capable of representing any value of any signed integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [INTMAX_MIN, INTMAX_MAX].

The macro INTMAX_C() expands its argument to an integer constant of type *intmax_t*.

The length modifier for *intmax_t* for the printf(3) and the scanf(3) families of functions is **j**; resulting commonly in %**jd** or %**ji** for printing *intmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *intmax_t* is not large enough to represent values of type __int128 in implementations where __int128 is defined and long long is less than 128 bits wide.

See also: the *uintmax_t* type in this page.

intN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

int8_t, *int16_t*, *int32_t*, *int64_t*

A signed integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [INTN_MIN, INTN_MAX], substituting N by the appropriate number.

According to POSIX, *int8_t*, *int16_t*, and *int32_t* are required; *int64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *intN_t* types for the printf(3) family of functions are expanded by macros of the forms **PRIIdN** and **PRIiN** (defined in <inttypes.h>); resulting for example in %"**PRI64**" or %"**PRIi64**" for printing *int64_t* values. The length modifiers for the *intN_t*

types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNdN** and **SCNiN**, (defined in *<inttypes.h>*); resulting for example in **%"SCNd8"** or **%"SCNi8"** for scanning *int8_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *uintN_t*, and *uintmax_t* types in this page.

intptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

A signed integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [**INTPTR_MIN**, **INTPTR_MAX**].

The length modifier for *intptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIIdPTR** and **PRIiPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIIdPTR"** or **%"PRIiPTR"** for printing *intptr_t* values. The length modifier for *intptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNdPTR** and **SCNiPTR**, (defined in *<inttypes.h>*); resulting commonly in **%"SCNdPTR"** or **%"SCNiPTR"** for scanning *intptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *uintptr_t* and *void ** types in this page.

lconv

Include: *<locale.h>*.

```
struct lconv {                                /* Values in the "C" locale: */
    char    *decimal_point;                  /* ". " */
    char    *thousands_sep;                 /* " " */
    char    *grouping;                      /* " " */
    char    *mon_decimal_point;             /* " " */
    char    *mon_thousands_sep;             /* " " */
    char    *mon_grouping;                  /* " " */
    char    *positive_sign;                 /* " " */
    char    *negative_sign;                 /* " " */
    char    *currency_symbol;               /* " " */
    char    frac_digits;                   /* CHAR_MAX */
    char    p_cs_precedes;                 /* CHAR_MAX */
    char    n_cs_precedes;                 /* CHAR_MAX */
    char    p_sep_by_space;                /* CHAR_MAX */
    char    n_sep_by_space;                /* CHAR_MAX */
    char    p_sign_posn;                   /* CHAR_MAX */
    char    n_sign_posn;                   /* CHAR_MAX */
    char    *int_curr_symbol;              /* " " */
    char    int_frac_digits;               /* CHAR_MAX */
    char    int_p_cs_precedes;             /* CHAR_MAX */
    char    int_n_cs_precedes;             /* CHAR_MAX */
    char    int_p_sep_by_space;             /* CHAR_MAX */
    char    int_n_sep_by_space;             /* CHAR_MAX */
    char    int_p_sign_posn;                /* CHAR_MAX */
    char    int_n_sign_posn;                /* CHAR_MAX */
};
```

Contains members related to the formatting of numeric values. In the "C" locale, its members have the values shown in the comments above.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **setlocale**(3), **localeconv**(3), **charsets**(5), **locale**(7)

ldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long      quot; /* Quotient */
    long      rem;   /* Remainder */
} ldiv_t;
```

It is the type of the value returned by the **ldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **ldiv(3)**

lldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long long  quot; /* Quotient */
    long long  rem;  /* Remainder */
} lldiv_t;
```

It is the type of the value returned by the **lldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **lldiv(3)**

off_t

Include: <sys/types.h>. Alternatively, <aio.h>, <fcntl.h>, <stdio.h>, <sys/mman.h>, <sys/stat.h>, or <unistd.h>.

Used for file sizes. According to POSIX, this shall be a signed integer type.

Versions: <aio.h> and <stdio.h> define *off_t* since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

Notes: On some architectures, the width of this type can be controlled with the feature test macro **_FILE_OFFSET_BITS**.

See also: **lseek(2)**, **mmap(2)**, **posix_fadvise(2)**, **pread(2)**, **truncate(2)**, **fseeko(3)**, **lockf(3)**, **posix_fallocate(3)**, **feature_test_macros(7)**

pid_t

Include: <sys/types.h>. Alternatively, <fcntl.h>, <sched.h>, <signal.h>, <spawn.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/wait.h>, <termios.h>, <time.h>, <unistd.h>, or <utmpx.h>.

This type is used for storing process IDs, process group IDs, and session IDs. According to POSIX, it shall be a signed integer type, and the implementation shall support one or more programming environments where the width of *pid_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: **fork(2)**, **getpid(2)**, **getppid(2)**, **getsid(2)**, **gettid(2)**, **getpgid(2)**, **kill(2)**, **pidfd_open(2)**, **sched_setscheduler(2)**, **waitpid(2)**, **sigqueue(3)**, **credentials(7)**,

ptrdiff_t

Include: <stddef.h>.

Used for a count of elements, and array indices. It is the result of subtracting two pointers. According to the C language standard, it shall be a signed integer type capable of storing values in the range [**PTRDIFF_MIN**, **PTRDIFF_MAX**].

The length modifier for *ptrdiff_t* for the **printf(3)** and the **scanf(3)** families of functions is **t**; resulting commonly in **%td** or **%ti** for printing *ptrdiff_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the `size_t` and `ssize_t` types in this page.

`regex_t`

Include: `<regex.h>`.

```
typedef struct {
    size_t re_nsub; /* Number of parenthesized subexpressions. */
} regex_t;
```

This is a structure type used in regular expression matching. It holds a compiled regular expression, compiled with `regcomp(3)`.

Conforming to: POSIX.1-2001 and later.

See also: `regex(3)`

`regmatch_t`

Include: `<regex.h>`.

```
typedef struct {
    regoff_t rm_so; /* Byte offset from start of string
                      to start of substring */
    regoff_t rm_eo; /* Byte offset from start of string of
                      the first character after the end of
                      substring */
} regmatch_t;
```

This is a structure type used in regular expression matching.

Conforming to: POSIX.1-2001 and later.

See also: `regexec(3)`

`regoff_t`

Include: `<regex.h>`.

According to POSIX, it shall be a signed integer type capable of storing the largest value that can be stored in either a `ptrdiff_t` type or a `ssize_t` type.

Versions: Prior to POSIX.1-2008, the type was capable of storing the largest value that can be stored in either an `off_t` type or a `ssize_t` type.

Conforming to: POSIX.1-2001 and later.

See also: the `regmatch_t` structure and the `ptrdiff_t` and `ssize_t` types in this page.

`sigevent`

Include: `<signal.h>`. Alternatively, `<aio.h>`, `<mqueue.h>`, or `<time.h>`.

```
struct sigevent {
    int           sigev_notify; /* Notification type */
    int           sigev_signo;  /* Signal number */
    union sigval  sigev_value; /* Signal value */
    void         (*sigev_notify_function)(union sigval);
                 /* Notification function */
    pthread_attr_t *sigev_notify_attributes;
                 /* Notification attributes */
};
```

For further details about this type, see `sigevent(7)`.

Versions: `<aio.h>` and `<time.h>` define `sigevent` since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

See also: `timer_create(2)`, `getaddrinfo_a(3)`, `lio_listio(3)`, `mq_notify(3)`

See also the *aiocb* structure in this page.

siginfo_t

Include: <signal.h>. Alternatively, <sys/wait.h>.

```
typedef struct {
    int      si_signo; /* Signal number */
    int      si_code;  /* Signal code */
    pid_t   si_pid;   /* Sending process ID */
    uid_t   si_uid;   /* Real user ID of sending process */
    void    *si_addr; /* Address of faulting instruction */
    int     si_status; /* Exit value or signal */
    union sigval si_value; /* Signal value */
} siginfo_t;
```

Information associated with a signal. For further details on this structure (including additional, Linux-specific fields), see **sigaction**(2).

Conforming to: POSIX.1-2001 and later.

See also: **pidfd_send_signal**(2), **rt_sigqueueinfo**(2), **sigaction**(2), **sigwaitinfo**(2), **psiginfo**(3)

sigset_t

Include: <signal.h>. Alternatively, <spawn.h>, or <sys/select.h>.

This is a type that represents a set of signals. According to POSIX, this shall be an integer or structure type.

Conforming to: POSIX.1-2001 and later.

See also: **epoll_pwait**(2), **ppoll**(2), **pselect**(2), **sigaction**(2), **signalfd**(2), **sigpending**(2), **sigprocmask**(2), **sigsuspend**(2), **sigwaitinfo**(2), **signal**(7)

sigval

Include: <signal.h>.

```
union sigval {
    int     sigval_int; /* Integer value */
    void   *sigval_ptr; /* Pointer value */
};
```

Data passed with a signal.

Conforming to: POSIX.1-2001 and later.

See also: **pthread_sigqueue**(3), **sigqueue**(3), **sigevent**(7)

See also the *sigevent* structure and the *siginfo_t* type in this page.

size_t

Include: <stddef.h> or <sys/types.h>. Alternatively, <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <monetary.h>, <mqueue.h>, <ndbm.h>, <pwd.h>, <regex.h>, <search.h>, <signal.h>, <stdio.h>, <stdlib.h>, <string.h>, <strings.h>, <sys/mman.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/socket.h>, <sys/uio.h>, <time.h>, <unistd.h>, <wchar.h>, or <wctype.h>.

Used for a count of bytes. It is the result of the *sizeof* operator. According to the C language standard, it shall be an unsigned integer type capable of storing values in the range [0, **SIZE_MAX**]. According to POSIX, the implementation shall support one or more programming environments where the width of *size_t* is no greater than the width of the type *long*.

The length modifier for *size_t* for the **printf**(3) and the **scanf**(3) families of functions is **z**; resulting commonly in **%zu** or **%zx** for printing *size_t* values.

Versions: <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <mqueue.h>, <pwd.h>, <signal.h>, and <sys/socket.h> define *size_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **read(2)**, **write(2)**, **fread(3)**, **fwrite(3)**, **memcmp(3)**, **memcpy(3)**, **memset(3)**, **off-setof(3)**

See also the *ptrdiff_t* and *ssize_t* types in this page.

ssize_t

Include: <sys/types.h>. Alternatively, <aio.h>, <monetary.h>, <mqueue.h>, <stdio.h>, <sys/msg.h>, <sys/socket.h>, <sys/uio.h>, or <unistd.h>.

Used for a count of bytes or an error indication. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, **SSIZE_MAX**], and the implementation shall support one or more programming environments where the width of *ssize_t* is no greater than the width of the type *long*.

Glibc and most other implementations provide a length modifier for *ssize_t* for the **printf(3)** and the **scanf(3)** families of functions, which is **z**; resulting commonly in **%zd** or **%zi** for printing *ssize_t* values. Although **z** works for *ssize_t* on most implementations, portable POSIX programs should avoid using it—for example, by converting the value to *intmax_t* and using its length modifier (**j**).

Conforming to: POSIX.1-2001 and later.

See also: **read(2)**, **readlink(2)**, **readv(2)**, **recv(2)**, **send(2)**, **write(2)**

See also the *ptrdiff_t* and *size_t* types in this page.

suseconds_t

Include: <sys/types.h>. Alternatively, <sys/select.h>, or <sys/time.h>.

Used for time in microseconds. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, 1000000], and the implementation shall support one or more programming environments where the width of *suseconds_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: the *timeval* structure in this page.

time_t

Include: <time.h> or <sys/types.h>. Alternatively, <sched.h>, <sys/msg.h>, <sys/select.h>, <sys/sem.h>, <sys/shm.h>, <sys/stat.h>, <sys/time.h>, or <utime.h>.

Used for time in seconds. According to POSIX, it shall be an integer type.

Versions: <sched.h> defines *time_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **stime(2)**, **time(2)**, **ctime(3)**, **difftime(3)**

timer_t

Include: <sys/types.h>. Alternatively, <time.h>.

Used for timer ID returned by **timer_create(2)**. According to POSIX, there are no defined comparison or assignment operators for this type.

Conforming to: POSIX.1-2001 and later.

See also: **timer_create(2)**, **timer_delete(2)**, **timer_getoverrun(2)**, **timer_settime(2)**

timespec

Include: <time.h>. Alternatively, <aio.h>, <mqueue.h>, <sched.h>, <signal.h>, <sys/select.h>, or <sys/stat.h>.

```
struct timespec {
    time_t tv_sec; /* Seconds */
```

```
    long      tv_nsec; /* Nanoseconds */
}
```

Describes times in seconds and nanoseconds.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **clock_gettime(2)**, **clock_nanosleep(2)**, **nanosleep(2)**, **timerfd_gettime(2)**, **timer_gettime(2)**

timeval

Include: <sys/time.h>. Alternatively, <sys/resource.h>, <sys/select.h>, or <utmpx.h>.

```
struct timeval {
    time_t      tv_sec;   /* Seconds */
    suseconds_t tv_usec; /* Microseconds */
};
```

Describes times in seconds and microseconds.

Conforming to: POSIX.1-2001 and later.

See also: **gettimeofday(2)**, **select(2)**, **utimes(2)**, **adjtime(3)**, **futimes(3)**, **timeradd(3)**

uid_t

Include: <sys/types.h>. Alternatively, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold user IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: **chown(2)**, **getuid(2)**, **geteuid(2)**, **getresuid(2)**, **getpwnam(2)**, **credentials(7)**

uintmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

An unsigned integer type capable of representing any value of any unsigned integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [0, **UINTMAX_MAX**].

The macro **UINTMAX_C()** expands its argument to an integer constant of type *uintmax_t*.

The length modifier for *uintmax_t* for the **printf(3)** and the **scanf(3)** families of functions is **j**; resulting commonly in **%ju** or **%jx** for printing *uintmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *uintmax_t* is not large enough to represent values of type *unsigned __int128* in implementations where *unsigned __int128* is defined and *unsigned long long* is less than 128 bits wide.

See also: the *intmax_t* type in this page.

uintN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

uint8_t, *uint16_t*, *uint32_t*, *uint64_t*

An unsigned integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [0, **UINTN_MAX**], substituting N by the appropriate number.

According to POSIX, *uint8_t*, *uint16_t*, and *uint32_t* are required; *uint64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *uintN_t* types for the **printf(3)** family of functions are expanded by macros of the forms **PRIuN**, **PRIoN**, **PRIxN**, and **PRIXN** (defined in <inttypes.h>); resulting for example in **%"PRIu32"** or **%"PRIx32"** for printing *uint32_t* values. The length modifiers for

the *uintN_t* types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNuN**, **SCNoN**, **SCNxN**, and **SCNXN** (defined in *<inttypes.h>*); resulting for example in **%"SCNu16"** or **%"SCNx16"** for scanning *uint16_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *intN_t*, and *uintmax_t* types in this page.

uintptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

An unsigned integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [0, **UINTPTR_MAX**].

The length modifier for *uintptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIupTR**, **PRIoPTR**, **PRIxPTR**, and **PRIxPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIupTR"** or **%"PRIxPTR"** for printing *uintptr_t* values. The length modifier for *uintptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNuPTR**, **SCNoPTR**, **SCNxPTR**, and **SCNXPTR** (defined in *<inttypes.h>*); resulting commonly in **%"SCNuPTR"** or **%"SCNxPTR"** for scanning *uintptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intptr_t* and *void ** types in this page.

va_list

Include: *<stdarg.h>*. Alternatively, *<stdio.h>*, or *<wchar.h>*.

Used by functions with a varying number of arguments of varying types. The function must declare an object of type *va_list* which is used by the macros **va_start**(3), **va_arg**(3), **va_copy**(3), and **va_end**(3) to traverse the list of arguments.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **va_start**(3), **va_arg**(3), **va_copy**(3), **va_end**(3)

*void **

According to the C language standard, a pointer to any object type may be converted to a pointer to *void* and back. POSIX further requires that any pointer, including pointers to functions, may be converted to a pointer to *void* and back.

Conversions from and to any other pointer type are done implicitly, not requiring casts at all. Note that this feature prevents any kind of type checking: the programmer should be careful not to convert a *void ** value to a type incompatible to that of the underlying data, because that would result in undefined behavior.

This type is useful in function parameters and return value to allow passing values of any type. The function will typically use some mechanism to know the real type of the data being passed via a pointer to *void*.

A value of this type can't be dereferenced, as it would give a value of type *void*, which is not possible. Likewise, pointer arithmetic is not possible with this type. However, in GNU C, pointer arithmetic is allowed as an extension to the standard; this is done by treating the size of a *void* or of a function as 1. A consequence of this is that *sizeof* is also allowed on *void* and on function types, and returns 1.

The conversion specifier for *void ** for the **printf**(3) and the **scanf**(3) families of functions is **p**.

Versions: The POSIX requirement about compatibility between *void ** and function pointers was added in POSIX.1-2008 Technical Corrigendum 1 (2013).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **malloc**(3), **memcmp**(3), **memcpy**(3), **memset**(3)

See also the *intptr_t* and *uintptr_t* types in this page.

NOTES

The structures described in this manual page shall contain, at least, the members shown in their definition, in no particular order.

Most of the integer types described in this page don't have a corresponding length modifier for the `printf(3)` and the `scanf(3)` families of functions. To print a value of an integer type that doesn't have a length modifier, it should be converted to `intmax_t` or `uintmax_t` by an explicit cast. To scan into a variable of an integer type that doesn't have a length modifier, an intermediate temporary variable of type `intmax_t` or `uintmax_t` should be used. When copying from the temporary variable to the destination variable, the value could overflow. If the type has upper and lower limits, the user should check that the value is within those limits, before actually copying the value. The example below shows how these conversions should be done.

Conventions used in this page

In "Conforming to" we only concern ourselves with C99 and later and POSIX.1-2001 and later. Some types may be specified in earlier versions of one of these standards, but in the interests of simplicity we omit details from earlier standards.

In "Include", we first note the "primary" header(s) that define the type according to either the C or POSIX.1 standards. Under "Alternatively", we note additional headers that the standards specify shall define the type.

EXAMPLES

The program shown below scans from a string and prints a value stored in a variable of an integer type that doesn't have a length modifier. The appropriate conversions from and to `intmax_t`, and the appropriate range checks, are used as explained in the notes section above.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int
main (void)
{
    static const char *const str = "500000 us in half a second";
    suseconds_t us;
    intmax_t tmp;

    /* Scan the number from the string into the temporary variable */

    sscanf(str, "%jd", &tmp);

    /* Check that the value is within the valid range of suseconds_t */

    if (tmp < -1 || tmp > 1000000) {
        fprintf(stderr, "Scanned value outside valid range!\n");
        exit(EXIT_FAILURE);
    }

    /* Copy the value to the suseconds_t variable 'us' */

    us = tmp;

    /* Even though suseconds_t can hold the value -1, this isn't
       a sensible number of microseconds */
}
```

```
if (us < 0) {
    fprintf(stderr, "Scanned value shouldn't be negative!\n");
    exit(EXIT_FAILURE);
}

/* Print the value */

printf("There are %jd microseconds in half a second.\n",
       (intmax_t) us);

exit(EXIT_SUCCESS);
}
```

SEE ALSO

[feature_test_macros\(7\)](#), [standards\(7\)](#)

COLPHON

This page is part of release 5.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.

NAME

renice – alter priority of running processes

SYNOPSIS

renice [**-n**] *priority* [**-g|-p|-u**] *identifier...*

DESCRIPTION

renice alters the scheduling priority of one or more running processes. The first argument is the *priority* value to be used. The other arguments are interpreted as process IDs (by default), process group IDs, user IDs, or user names. **renice**'ing a process group causes all processes in the process group to have their scheduling priority altered. **renice**'ing a user causes all processes owned by the user to have their scheduling priority altered.

OPTIONS

-n, --priority *priority*

Specify the scheduling *priority* to be used for the process, process group, or user. Use of the option **-n** or **--priority** is optional, but when used it must be the first argument.

-g, --pgrp

Interpret the succeeding arguments as process group IDs.

-p, --pid

Interpret the succeeding arguments as process IDs (the default).

-u, --user

Interpret the succeeding arguments as usernames or UIDs.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

FILES

/etc/passwd

to map user names to user IDs

NOTES

Users other than the superuser may only alter the priority of processes they own. Furthermore, an unprivileged user can only *increase* the "nice value" (i.e., choose a lower priority) and such changes are irreversible unless (since Linux 2.6.12) the user has a suitable "nice" resource limit (see **ulimit(1p)** and **getrlimit(2)**).

The superuser may alter the priority of any process and set the priority to any value in the range -20 to 19. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to), 0 (the "base" scheduling priority), anything negative (to make things go very fast).

HISTORY

The **renice** command appeared in 4.0BSD.

EXAMPLES

The following command would change the priority of the processes with PIDs 987 and 32, plus all processes owned by the users daemon and root:

```
renice +1 987 -u daemon root -p 32
```

SEE ALSO

nice(1), **chrt(1)**, **getpriority(2)**, **setpriority(2)**, **credentials(7)**, **sched(7)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The `renice` command is part of the `util-linux` package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

rfkill – tool for enabling and disabling wireless devices

SYNOPSIS

rfkill [options] [*command*] [*ID|type ...*]

DESCRIPTION

rfkill lists, enabling and disabling wireless devices.

The command "list" output format is deprecated and maintained for backward compatibility only. The new output format is the default when no command is specified or when the option **--output** is used.

The default output is subject to change. So whenever possible, you should avoid using default outputs in your scripts. Always explicitly define expected columns by using the **--output** option together with a columns list in environments where a stable output is required.

OPTIONS

-J, --json

Use JSON output format.

-n, --noheadings

Do not print a header line.

-o, --output

Specify which output columns to print. Use **--help** to get a list of available columns.

--output-all

Output all available columns.

-r, --raw

Use the raw output format.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

COMMANDS

help

Display help text and exit.

event

Listen for rfkill events and display them on stdout.

list [*id|type ...*]

List the current state of all available devices. The command output format is deprecated, see the **DESCRIPTION** section. It is a good idea to check with **list** command *id* or *type* scope is appropriate before setting **block** or **unblock**. Special *all* type string will match everything. Use of multiple *ID* or *type* arguments is supported.

block *id|type [...]*

Disable the corresponding device.

unblock *id|type [...]*

Enable the corresponding device. If the device is hard-blocked, for example via a hardware switch, it will remain unavailable though it is now soft-unblocked.

toggle *id|type* [...]
Enable or disable the corresponding device.

EXAMPLE

```
rfkill --output ID,TYPE  
rfkill block all  
rfkill unblock wlan  
rfkill block bluetooth uwb wimax wwan gps fm nfc
```

AUTHORS

rfkill was originally written by [Johannes Berg](#) <johannes@sipsolutions.net> and [Marcel Holtmann](#) <marcel@holtmann.org>. The code has been later modified by [Sami Kerola](#) <kerolasa@iki.fi> and [Karel Zak](#) <kzak@redhat.com> for the util-linux project.

This manual page was written by [Darren Salt](#) <linux@youmustbejoking.demon.co.uk> for the Debian project (and may be used by others).

SEE ALSO

[powertop\(8\)](#), [systemd-rfkill\(8\)](#), [Linux kernel documentation](#)
<<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/driver-api/rfkill.rst>>

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **rfkill** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](#) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

rm – remove files or directories

SYNOPSIS

rm [*OPTION*]... [*FILE*]...

DESCRIPTION

This manual page documents the GNU version of **rm**. **rm** removes each specified file. By default, it does not remove directories.

If the **-I** or **--interactive=once** option is given, and there are more than three files or the **-r**, **-R**, or **--recursive** are given, then **rm** prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

Otherwise, if a file is unwritable, standard input is a terminal, and the **-f** or **--force** option is not given, or the **-i** or **--interactive=always** option is given, **rm** prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

OPTIONS

Remove (unlink) the FILE(s).

-f, --force

ignore nonexistent files and arguments, never prompt

-i

prompt before every removal

-I

prompt once before removing more than three files, or when removing recursively; less intrusive than **-i**, while still giving protection against most mistakes

--interactive[=*WHEN*]

prompt according to *WHEN*: never, once (**-I**), or always (**-i**); without *WHEN*, prompt always

--one-file-system

when removing a hierarchy recursively, skip any directory that is on a file system different from that of the corresponding command line argument

--no-preserve-root

do not treat '/' specially

--preserve-root[=*all*]

do not remove '/' (default); with 'all', reject any command line argument on a separate device from its parent

-r, -R, --recursive

remove directories and their contents recursively

-d, --dir

remove empty directories

-v, --verbose

explain what is being done

--help

display this help and exit

--version

output version information and exit

By default, **rm** does not remove directories. Use the **--recursive** (**-r** or **-R**) option to remove each listed directory, too, along with all of its contents.

To remove a file whose name starts with a '**-**', for example '**-foo**', use one of these commands:

```
rm -- -foo
```

```
rm ./-foo
```

Note that if you use **rm** to remove a file, it might be possible to recover some of its contents, given sufficient expertise and/or time. For greater assurance that the contents are truly unrecoverable, consider using **shred**.

AUTHOR

Written by Paul Rubin, David MacKenzie, Richard M. Stallman, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`unlink(1)`, `unlink(2)`, `chattr(1)`, `shred(1)`

Full documentation <<https://www.gnu.org/software/coreutils/rm>>
or available locally via: `info '(coreutils) rm invocation'`

NAME

rmdir – remove empty directories

SYNOPSIS

rmdir [*OPTION*]... *DIRECTORY*...

DESCRIPTION

Remove the *DIRECTORY*(ies), if they are empty.

--ignore-fail-on-non-empty

ignore each failure that is solely because a directory
is non-empty

-p, --parents

remove *DIRECTORY* and its ancestors; e.g., '**rmdir -p a/b/c**' is similar to '**rmdir a/b/c a/b a**'

-v, --verbose

output a diagnostic for every directory processed

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent
permitted by law.

SEE ALSO

rmdir(2)

Full documentation <<https://www.gnu.org/software/coreutils/rmdir>>
or available locally via: info '(coreutils) rmdir invocation'

NAME

rmdir – delete a directory

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <unistd.h>
int rmdir(const char *pathname);
```

DESCRIPTION

rmdir() deletes a directory, which must be empty.

RETURN VALUE

On success, zero is returned. On error, **-1** is returned, and *errno* is set to indicate the error.

ERRORS**EACCES**

Write access to the directory containing *pathname* was not allowed, or one of the directories in the path prefix of *pathname* did not allow search permission. (See also **path_resolution(7)**.)

EBUSY

pathname is currently in use by the system or some process that prevents its removal. On Linux, this means *pathname* is currently used as a mount point or is the root directory of the calling process.

EFAULT

pathname points outside your accessible address space.

EINVAL

pathname has **.** as last component.

ELOOP

Too many symbolic links were encountered in resolving *pathname*.

ENAMETOOLONG

pathname was too long.

ENOENT

A directory component in *pathname* does not exist or is a dangling symbolic link.

ENOMEM

Insufficient kernel memory was available.

ENOTDIR

pathname, or a component used as a directory in *pathname*, is not, in fact, a directory.

ENOTEMPTY

pathname contains entries other than **.** and **..**; or, *pathname* has **..** as its final component. POSIX.1 also allows **EXIST** for this condition.

EPERM

The directory containing *pathname* has the sticky bit (**S_ISVTX**) set and the process's effective user ID is neither the user ID of the file to be deleted nor that of the directory containing it, and the process is not privileged (Linux: does not have the **CAP_FOWNER** capability).

EPEERM

The filesystem containing *pathname* does not support the removal of directories.

EROFS

pathname refers to a directory on a read-only filesystem.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD.

BUGS

Infelicities in the protocol underlying NFS can cause the unexpected disappearance of directories which are still being used.

SEE ALSO

rm(1), rmdir(1), chdir(2), chmod(2), mkdir(2), rename(2), unlink(2), unlinkat(2)

NAME

rmmod – Simple program to remove a module from the Linux Kernel

SYNOPSIS

rmmod [**-f**] [**-s**] [**-v**] [*modulename*]

DESCRIPTION

rmmod is a trivial program to remove a module (when module unloading support is provided) from the kernel. Most users will want to use **modprobe(8)** with the **-r** option instead.

OPTIONS**-v, --verbose**

Print messages about what the program is doing. Usually **rmmod** prints messages only if something goes wrong.

-f, --force

This option can be extremely dangerous: it has no effect unless **CONFIG_MODULE_FORCE_UNLOAD** was set when the kernel was compiled. With this option, you can remove modules which are being used, or which are not designed to be removed, or have been marked as unsafe (see **lsmod(8)**).

-s, --syslog

Send errors to syslog instead of standard error.

-V --version

Show version of program and exit.

COPYRIGHT

This manual page originally Copyright 2002, Rusty Russell, IBM Corporation. Maintained by Jon Masters and others.

SEE ALSO

modprobe(8), **insmod(8)**, **lsmod(8)**, **modinfo(8)** **depmod(8)**

AUTHORS

Jon Masters <jcm@jonmasters.org>
Developer

Lucas De Marchi <lucas.de.marchi@gmail.com>
Developer

NAME

route – show / manipulate the IP routing table

SYNOPSIS

```
route [-CFvnNee] [-A family |-4|-6]
route  [-v] [-A family |-4|-6] add [-net|-host] target [netmask Nm] [gw Gw] [metric N] [mss M]
       [window W] [irtt I] [reject] [mod] [dyn] [reinstate] [[dev] If]
route   [-v] [-A family |-4|-6] del [-net|-host] target [gw Gw] [netmask Nm] [metric M] [[dev] If]
route   [-V] [--version] [-h] [--help]
```

DESCRIPTION

Route manipulates the kernel's IP routing tables. Its primary use is to set up static routes to specific hosts or networks via an interface after it has been configured with the **ifconfig(8)** program.

When the **add** or **del** options are used, **route** modifies the routing tables. Without these options, **route** displays the current contents of the routing tables.

OPTIONS**-A** *family*

use the specified address family (eg ‘inet’). Use **route --help** for a full list. You can use **-6** as an alias for **--inet6** and **-4** as an alias for **-A inet**

-F operate on the kernel's FIB (Forwarding Information Base) routing table. This is the default.

-C operate on the kernel's routing cache.

-v select verbose operation.

-n show numerical addresses instead of trying to determine symbolic host names. This is useful if you are trying to determine why the route to your nameserver has vanished.

-e use **netstat(8)**-format for displaying the routing table. **-ee** will generate a very long line with all parameters from the routing table.

del delete a route.

add add a new route.

target the destination network or host. You can provide an addresses or symbolic network or host name. Optionally you can use */prefixlen* notation instead of using the **netmask** option.

-net the *target* is a network.

-host the *target* is a host.

netmask *NM*

when adding a network route, the netmask to be used.

gw *GW* route packets via a gateway.

NOTE: The specified gateway must be reachable first. This usually means that you have to set up a static route to the gateway beforehand. If you specify the address of one of your local interfaces, it will be used to decide about the interface to which the packets should be routed to. This is a BS-Dism compatibility hack.

metric *M*

set the metric field in the routing table (used by routing daemons) to *M*. If this option is not specified the metric for inet6 (IPv6) address family defaults to '1', for inet (IPv4) it defaults to '0'. You should always specify an explicit metric value to not rely on those defaults - they also differ from iproute2.

mss *M* sets MTU (Maximum Transmission Unit) of the route to *M* bytes. Note that the current implementation of the route command does not allow the option to set the Maximum Segment Size (MSS).

window *W*

set the TCP window size for connections over this route to *W* bytes. This is typically only used on AX.25 networks and with drivers unable to handle back to back frames.

irtt *I* set the initial round trip time (irtt) for TCP connections over this route to *I* milliseconds (1-12000). This is typically only used on AX.25 networks. If omitted the RFC 1122 default of 300ms is used.

reject install a blocking route, which will force a route lookup to fail. This is for example used to mask out networks before using the default route. This is NOT for firewalling.

mod, dyn, reinstate

install a dynamic or modified route. These flags are for diagnostic purposes, and are generally only set by routing daemons.

dev *If* force the route to be associated with the specified device, as the kernel will otherwise try to determine the device on its own (by checking already existing routes and device specifications, and where the route is added to). In most normal networks you won't need this.

If **dev *If*** is the last option on the command line, the word **dev** may be omitted, as it's the default. Otherwise the order of the route modifiers (**metric netmask gw dev**) doesn't matter.

EXAMPLES

route add -net 127.0.0.0 netmask 255.0.0.0 metric 1024 dev lo

adds the normal loopback entry, using netmask 255.0.0.0 and associated with the "lo" device (assuming this device was previously set up correctly with **ifconfig(8)**).

route add -net 192.56.76.0 netmask 255.255.255.0 metric 1024 dev eth0

adds a route to the local network 192.56.76.x via "eth0". The word "dev" can be omitted here.

route del default

deletes the current default route, which is labeled "default" or 0.0.0.0 in the destination field of the current routing table.

route del -net 192.56.76.0 netmask 255.255.255.0

deletes the route. Since the Linux routing kernel uses classless addressing, you pretty much always have to specify the netmask that is same as as seen in 'route -n' listing.

route add default gw mango

adds a default route (which will be used if no other route matches). All packets using this route will be gatewayed through the address of a node named "mango". The device which will actually be used for that route depends on how we can reach "mango" - "mango" must be on directly reachable route.

route add mango sl0

Adds the route to the host named "mango" via the SLIP interface (assuming that "mango" is the SLIP host).

route add -net 192.57.66.0 netmask 255.255.255.0 gw mango

This command adds the net "192.57.66.x" to be gatewayed through the former route to the SLIP interface.

route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0

This is an obscure one documented so people know how to do it. This sets all of the class D (multicast) IP routes to go via "eth0". This is the correct normal configuration line with a multicasting kernel.

route add -net 10.0.0.0 netmask 255.0.0.0 metric 1024 reject

This installs a rejecting route for the private network "10.x.x.x."

route -6 add 2001:0002::/48 metric 1 dev eth0

This adds a IPv6 route with the specified metric to be directly reachable via eth0.

OUTPUT

The output of the kernel routing table is organized in the following columns

Destination

The destination network or destination host.

Gateway

The gateway address or '*' if none set.

Genmask

The netmask for the destination net; '255.255.255.255' for a host destination and '0.0.0.0' for the **default** route.

Flags

Possible flags include
U (route is **up**)
H (target is a **host**)
G (use **gateway**)
R (**reinstate** route for dynamic routing)
D (**dynamically** installed by daemon or redirect)
M (**modified** from routing daemon or redirect)
A (installed by **addrconf**)
C (**cache** entry)
! (**reject** route)

Metric

The 'distance' to the target (usually counted in hops).

Ref

Number of references to this route. (Not used in the Linux kernel.)

Use

Count of lookups for the route. Depending on the use of -F and -C this will be either route cache misses (-F) or hits (-C).

Iface

Interface to which packets for this route will be sent.

MSS

Default maximum segment size for TCP connections over this route.

Window

Default window size for TCP connections over this route.

irtt

Initial RTT (Round Trip Time). The kernel uses this to guess about the best TCP protocol parameters without waiting on (possibly slow) answers.

HH (cached only)

The number of ARP entries and cached routes that refer to the hardware header cache for the cached route. This will be -1 if a hardware address is not needed for the interface of the cached route (e.g. lo).

Arp (cached only)

Whether or not the hardware address for the cached route is up to date.

FILES

*/proc/net/ipv6_route
/proc/net/route
/proc/net/rt_cache*

SEE ALSO

ifconfig(8), netstat(8), arp(8), rarp(8), ip(8)

HISTORY

Route for Linux was originally written by Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org> and then modified by Johannes Stille and Linus Torvalds for p15. Alan Cox added the mss and window options for Linux 1.1.22. irtt support and merged with netstat from Bernd Eckenfels.

AUTHOR

Currently maintained by Phil Blundell <Philip.Blundell@pobox.com> and Bernd Eckenfels <net-tools@lina.inka.de>.

rpm(8) - Linux man page

Name

rpm - RPM Package Manager

Synopsis

Querying and Verifying Packages:

rpm {-q|--query} [select-options] [query-options]

rpm {-V|--verify} [select-options] [verify-options]

rpm --import *PUBKEY* ...

rpm {-K|--checksig} [--nosignature] [--nodigest] *PACKAGE_FILE* ...

Installing, Upgrading, and Removing Packages:

rpm {-i|--install} [install-options] *PACKAGE_FILE* ...

rpm {-U|--upgrade} [install-options] *PACKAGE_FILE* ...

rpm {-F|--freshen} [install-options] *PACKAGE_FILE* ...

rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--notriggers] [--test] *PACKAGE_NAME* ...

Miscellaneous:

rpm {--initdb|--rebuilddb}

rpm {--addsign|--resign} *PACKAGE_FILE*...

rpm {--querytags|--showrc}

rpm {--setperms|--setugids} *PACKAGE_NAME* ...

select-options

**[*PACKAGE_NAME*] [-a,--all] [-f,--file *FILE*]
[-g,--group *GROUP*] {-p,--package *PACKAGE_FILE*}
[--fileid *ID*] [--hdrid *SHA1*] [--pkgid *MD5*] [--tid *TID*]
[--querybysize *HDRNUM*] [--triggeredby *PACKAGE_NAME*]
[--whatprovides *CAPABILITY*] [--whatrequires *CAPABILITY*]**

query-options

```
[--changelog] [-c,--configfiles] [-d,--docfiles] [--dump]
[--filesbypkg] [-i,--info] [--last] [-l,--list]
[--provides] [--qf,--queryformat QUERYFMT]
[-R,--requires] [--scripts] [-s,--state]
[--triggers,--triggerscripts]
```

verify-options

```
[--nodeps] [--nofiles] [--noscripts]
[--nodigest] [--nosignature]
[--nolinkto] [--nofiledigest] [--nosize] [--nouser]
[--nogroup] [--nomtime] [--nomode] [--nordev]
[--nocaps]
```

install-options

```
[--aid] [--allfiles] [--badreloc] [--excludedpath OLDPATH]
[--excludedocs] [--force] [-h,--hash]
[--ignoresize] [--ignorearch] [--ignoreos]
[--includedocs] [--justdb] [--nodeps]
[--nodigest] [--nosignature] [--nosuggest]
[--noorder] [--noscripts] [--notriggers]
[--oldpackage] [--percent] [--prefix NEWPATH]
[--relocate OLDPATH=NEWPATH]
[--replacefiles] [--replacepkgs]
[--test]
```

Description

rpm is a powerful **Package Manager**, which can be used to build, install, query, verify, update, and erase individual software packages. A **package** consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the package. **Packages** come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

One of the following basic modes must be selected: **Query**, **Verify**, **Signature Check**, **Install/Upgrade/Freshen**, **Uninstall**, **Initialize Database**, **Rebuild Database**, **Resign**, **Add Signature**, **Set Owners/Groups**, **Show Querytags**, and **Show Configuration**.

General Options

These options can be used in all the different modes.

-?, --help

Print a longer usage message than normal.

--version

Print a single line containing the version number of **rpm** being used.

--quiet

Print as little as possible - normally only error messages will be displayed.

-v

Print verbose information - normally routine progress messages will be displayed.

-vv

Print lots of ugly debugging information.

--rcfile FILELIST

Each of the files in the colon separated *FILELIST* is read sequentially by **rpm** for configuration information. Only the first file in the list must exist, and tildes will be expanded to the value of **\$HOME**. The default *FILELIST* is

/usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:/etc/rpmrc:~/.rpmrc.

--pipe CMD

Pipes the output of **rpm** to the command *CMD*.

--dbpath DIRECTORY

Use the database in *DIRECTORY* rather than the default path */var/lib/rpm*

--root DIRECTORY

Use the file system tree rooted at *DIRECTORY* for all operations. Note that this means the database within *DIRECTORY* will be used for dependency checks and any **scriptlet**(s) (e.g. **%post** if installing, or **%prep** if building, a package) will be run after a **chroot**(2) to *DIRECTORY*.

-D, --define='MACRO EXPR'

Defines *MACRO* with value *EXPR*.

-E, --eval='EXPR'

Prints macro expansion of *EXPR*.

Install and Upgrade Options

In these options, *PACKAGE_FILE* can be either **rpm** binary file or ASCII package manifest (see **PACKAGE SELECTION OPTIONS**), and may be specified as an **ftp** or **http** URL, in which case the package will be downloaded before being installed. See **FTP/HTTP OPTIONS** for information on **rpm**'s internal **ftp** and **http** client support.

The general form of an rpm install command is

rpm {-i|--install} [install-options] PACKAGE_FILE ...

This installs a new package.

The general form of an rpm upgrade command is

rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...

This upgrades or installs the package currently installed to a newer version. This is the same as install, except all other **version**(s) of the package are removed after the new package is installed.

rpm {-F|--freshen} [install-options] PACKAGE_FILE ...

This will upgrade packages, but only ones for which an earlier version is installed.

--aid

Add suggested packages to the transaction set when needed.

--allfiles

Installs or upgrades all the missingok files in the package, regardless if they exist.

--badreloc

Used with **--relocate**, permit relocations on all file paths, not just those *OLDPATH*'s included in the binary package relocation **hint**(s).

--excludepath *OLDPATH*

Don't install files whose name begins with *OLDPATH*.

--excludedocs

Don't install any files which are marked as documentation (which includes man pages and texinfo documents).

--force

Same as using **--replacepkgs**, **--replacefiles**, and **--oldpackage**.

-h, --hash

Print 50 hash marks as the package archive is unpacked. Use with **-v|--verbose** for a nicer display.

--ignoresize

Don't check mount file systems for sufficient disk space before installing this package.

--ignorearch

Allow installation or upgrading even if the architectures of the binary package and host don't match.

--ignoreos

Allow installation or upgrading even if the operating systems of the binary package and host don't match.

--includedocs

Install documentation files. This is the default behavior.

--justdb

Update only the database, not the filesystem.

--nodigest

Don't verify package or header digests when reading.

--nomanifest

Don't process non-package files as manifests.

--nosignature

Don't verify package or header signatures when reading.

--nodeps

Don't do a dependency check before installing or upgrading a package.

--nosuggest

Don't suggest **package**(s) that provide a missing dependency.

--noorder

Don't reorder the packages for an install. The list of packages would normally be reordered to satisfy dependencies.

--noscripts**--nopre****--nopost****--nopreun****--nopostun**

Don't execute the scriptlet of the same name. The **--noscripts** option is equivalent to

--nopre --nopost --nopreun

-

-

nopostun

and turns off the execution of the corresponding **%pre**, **%post**, **%preun**, and **%postun** **scriptlet**(s).

--notriggers**--notriggerin****--notriggerun****--notriggerpostun**

Don't execute any trigger scriptlet of the named type. The **--notriggers** option is equivalent to

--notriggerin --notriggerun --notriggerpostun

and turns off execution of the corresponding **%triggerin**, **%triggerun**, and **%triggerpostun** **scriptlet**(s).

--oldpackage

Allow an upgrade to replace a newer package with an older one.

--percent

Print percentages as files are unpacked from the package archive. This is intended to make **rpm** easy to run from other tools.

--prefix *NEWPATH*

For relocatable binary packages, translate all file paths that start with the installation prefix in the package relocation **hint**(s) to *NEWPATH*.

--relocate *OLDPATH=NEWPATH*

For relocatable binary packages, translate all file paths that start with *OLDPATH* in the package relocation **hint**(s) to *NEWPATH*. This option can be used repeatedly if several *OLDPATH*'s in the package are to be relocated.

--replacefiles

Install the packages even if they replace files from other, already installed, packages.

--replacepkgs

Install the packages even if some of them are already installed on this system.

--test

Do not install the package, simply check for and report potential conflicts.

Erase Options

The general form of an rpm erase command is

```
rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--notriggers] [--test]  
PACKAGE_NAME...
```

The following options may also be used:

--allmatches

Remove all versions of the package which match *PACKAGE_NAME*. Normally an error is issued if *PACKAGE_NAME* matches multiple packages.

--nodeps

Don't check dependencies before uninstalling the packages.

--noscripts

--nopreun

--nopostun

Don't execute the scriptlet of the same name. The **--noscripts** option during package erase is equivalent

to

--nopreun --nopostun

and turns off the execution of the corresponding **%preun**, and **%postun** **scriptlet**(s).

--notriggers

--notriggerun

--notriggerpostun

Don't execute any trigger scriptlet of the named type. The **--notriggers** option

is equivalent to

--notriggerun --notriggerpostun

and turns off execution of the corresponding **%triggerun**, and **%triggerpostun scriptlet(s)**.

--test

Don't really uninstall anything, just go through the motions. Useful in conjunction with the **-vv** option for debugging.

Query Options

The general form of an rpm query command is

rpm {-q|--query} [select-options] [query-options]

You may specify the format that package information should be printed in. To do this, you use the

--qf|--queryformat QUERYFMT

option, followed by the *QUERYFMT* format string. Query formats are modified versions of the standard ***printf(3)*** formatting. The format is made up of static strings (which may include standard C character escapes for newlines, tabs, and other special characters) and ***printf(3)*** type formatters. As **rpm** already knows the type to print, the type specifier must be omitted however, and replaced by the name of the header tag to be printed, enclosed by **{}** characters. Tag names are case insensitive, and the leading **RPMTAG_** portion of the tag name may be omitted as well.

Alternate output formats may be requested by following the tag with **:typetag**. Currently, the following types are supported:

:armor

Wrap a public key in ASCII armor.

:arraysize

Display number of elements in array tags.

:base64

Encode binary data using base64.

:date

Use ***strftime(3)*** "%c" format.

:day

Use ***strftime(3)*** "%a %b %d %Y" format.

:depflags

Format dependency comparison operator.

:deptype

Format dependency type.

:fflags

Format file flags.

:fstate

Format file state.

:hex

Format in hexadecimal.

:octal

Format in octal.

:perms

Format file permissions.

:pgpsig

Display signature fingerprint and time.

:shescape

Escape single quotes for use in a script.

:triggertype

Display trigger suffix.

:vflags

File verification flags.

:xml

Wrap data in simple xml markup.

For example, to print only the names of the packages queried, you could use **%{NAME}** as the format string. To print the packages name and distribution information in two columns, you could use **%-30{NAME}%{DISTRIBUTION}**. **rpm** will print a list of all of the tags it knows about when it is invoked with the **--querytags** argument.

There are two subsets of options for querying: package selection, and information selection.

Package Selection Options:

PACKAGE_NAME

Query installed package named *PACKAGE_NAME*.

-a, --all

Query all installed packages.

-f, --file FILE

Query package owning *FILE*.

--fileid ID

Query package that contains a given file identifier. The *ID* is the digest of the file contents. For different packages different hash algorithms may have been used (*MD5*, *SHA1*, *SHA256*, *SHA384*, *SHA512*, ...)

-g, --group GROUP

Query packages with the group of *GROUP*.

--hdrid SHA1

Query package that contains a given header identifier, i.e. the *SHA1* digest of the immutable header region.

-p, --package *PACKAGE_FILE*

Query an (uninstalled) package *PACKAGE_FILE*. The *PACKAGE_FILE* may be specified as an **ftp** or **http** style URL, in which case the package header will be downloaded and queried. See **FTP/HTTP OPTIONS** for information on rpm's internal **ftp** and **http** client support. The *PACKAGE_FILE* argument(s), if not a binary package, will be interpreted as an ASCII package manifest unless **--nomanifest** option is used. In manifests, comments are permitted, starting with a '#', and each line of a package manifest file may include white space separated glob expressions, including URL's, that will be expanded to paths that are substituted in place of the package manifest as additional *PACKAGE_FILE* arguments to the query.

--pkgid *MD5*

Query package that contains a given package identifier, i.e. the *MD5* digest of the combined header and payload contents.

--querybynumber *HDRNUM*

Query the *HDRNUM*th database entry directly; this is useful only for debugging.

--specfile *SPECFILE*

Parse and query *SPECFILE* as if it were a package. Although not all the information (e.g. file lists) is available, this type of query permits rpm to be used to extract information from spec files without having to write a specfile parser.

--tid *TID*

Query **package**(s) that have a given *TID* transaction identifier. A unix time stamp is currently used as a transaction identifier. All **package**(s) installed or erased within a single transaction have a common identifier.

--triggeredby *PACKAGE_NAME*

Query packages that are triggered by **package**(s) *PACKAGE_NAME*.

--whatprovides *CAPABILITY*

Query all packages that provide the *CAPABILITY* capability.

--whatrequires *CAPABILITY*

Query all packages that require *CAPABILITY* for proper functioning.

Package Query Options:

--changelog

Display change information for the package.

-c, --configfiles

List only configuration files (implies **-I**).

-d, --docfiles

List only documentation files (implies **-I**).

--dump

Dump file information as follows (implies **-I**):

path size mtime digest mode owner group isconfig isdoc rdev symlink

--filesbypkg

List all the files in each selected package.

-i, --info

Display package information, including name, version, and description. This uses the **-queryformat** if one was specified.

--last

Orders the package listing by install time such that the latest packages are at the top.

-l, --list

List files in package.

--provides

List capabilities this package provides.

-R, --requires

List capabilities on which this package depends.

--scripts

List the package specific **scriptlet**(s) that are used as part of the installation and uninstallation processes.

-s, --state

Display the *states* of files in the package (implies **-l**). The state of each file is one of *normal*, *not installed*, or *replaced*.

--triggers, --triggerscripts

Display the trigger scripts, if any, which are contained in the package.

Verify Options

The general form of an rpm verify command is

rpm {-V|--verify} [select-options] [verify-options]

Verifying a package compares information about the installed files in the package with information about the files taken from the package metadata stored in the rpm database. Among other things, verifying compares the size, digest, permissions, type, owner and group of each file. Any discrepancies are displayed. Files that were not installed from the package, for example, documentation files excluded on installation using the "**--excludedocs**" option, will be silently ignored.

The package selection options are the same as for package querying (including package manifest files as arguments). Other options unique to verify mode are:

--nodeps

Don't verify dependencies of packages.

--nodigest

Don't verify package or header digests when reading.

--nofiles

Don't verify any attributes of package files.

--noscripts

Don't execute the **%verifyscript** scriptlet (if any).

--nosignature

Don't verify package or header signatures when reading.

--nolinkto

--nofiledigest (formerly --nomd5)

--nosize

--nouser

--nogroup

--nomtime

--nomode

--nordev

Don't verify the corresponding file attribute.

The format of the output is a string of 8 characters, a possible attribute marker:

```
c %config configuration file.
d %doc documentation file.
g %ghost file (i.e. the file contents are not included in the package payload).
l %license license file.
r %readme readme file.
```

from the package header, followed by the file name. Each of the 8 characters denotes the result of a comparison of **attribute(s)** of the file to the value of those **attribute(s)** recorded in the database. A single "." (period) means the test passed, while a single "?" (question mark) indicates the test could not be performed (e.g. file permissions prevent reading). Otherwise, the (mnemonically em**Boldened**) character denotes failure of the corresponding **--verify** test:

S file Size differs

M Mode differs (includes permissions and file type)

5 digest (formerly MD5 sum) differs

D Device major/minor number mismatch

L readlink(2) path mismatch

U User ownership differs

G Group ownership differs

T mTime differs

P caPabilities differ

Digital Signature and Digest Verification

The general forms of rpm digital signature commands are

rpm --import PUBKEY ...

rpm {--checksig} [--nosignature] [--nodigest] PACKAGE_FILE ...

The **--checksig** option checks all the digests and signatures contained in *PACKAGE_FILE* to ensure the integrity and origin of the package. Note that signatures are now verified whenever a package is read, and **--checksig** is useful to verify all of the digests and signatures associated with a package.

Digital signatures cannot be verified without a public key. An ASCII armored public key can be added to the **rpm** database using **--import**. An imported public key is carried in a header, and key ring management is performed exactly like package management. For example, all currently imported public keys can be displayed by:

rpm -qa gpg-pubkey*

Details about a specific public key, when imported, can be displayed by querying. Here's information about the Red Hat GPG/DSA key:

rpm -qi gpg-pubkey-db42a60e

Finally, public keys can be erased after importing just like packages. Here's how to remove the Red Hat GPG/DSA key

rpm -e gpg-pubkey-db42a60e

Signing a Package

rpm --addsign|--resign PACKAGE_FILE ...

Both of the **--addsign** and **--resign** options generate and insert new signatures for each package *PACKAGE_FILE* given, replacing any existing signatures. There are two options for historical reasons, there is no difference in behavior currently.

Using Gpg to Sign Packages

In order to sign packages using GPG, **rpm** must be configured to run GPG and be able to find a key ring with the appropriate keys. By default, **rpm** uses the same conventions as GPG to find key rings, namely the **\$GNUPGHOME** environment variable. If your key rings are not located where GPG expects them to be, you will need to configure the macro **%_gpg_path** to be the location of the GPG key rings to use.

For compatibility with older versions of GPG, PGP, and rpm, only V3 OpenPGP signature packets should be configured. Either DSA or RSA verification algorithms can be used, but DSA is preferred.

If you want to be able to sign packages you create yourself, you also need to create your own public and secret key pair (see the GPG manual). You will also need to configure the

rpm macros**%_signature**

The signature type. Right now only gpg and pgp are supported.

%_gpg_name

The name of the "user" whose key you wish to use to sign your packages.

For example, to be able to use GPG to sign packages as the user "*John Doe <jdoe@foo.com>*" from the key rings located in `/etc/rpm/.gpg` using the executable `/usr/bin/gpg` you would include

```
%_signature gpg
%_gpg_path /etc/rpm/.gpg
%_gpg_name John Doe <jdoe@foo.com>
%_gpg /usr/bin/gpg
```

in a macro configuration file. Use `/etc/rpm/macros` for per-system configuration and `~/.rpmmacros` for per-user configuration. Typically it's sufficient to set just `%_gpg_name`.

Rebuild Database Options

The general form of an rpm rebuild database command is

rpm {--initdb|--rebuilddb} [-v] [--dbpath DIRECTORY] [--root DIRECTORY]

Use **--initdb** to create a new database if one doesn't already exist (existing database is not overwritten), use **--rebuilddb** to rebuild the database indices from the installed package headers.

Miscellaneous Commands**rpm --showrc**

shows the values **rpm** will use for all of the options are currently set in `rpmrc` and `macros` configuration **file(s)**.

rpm --setperms PACKAGE_NAME

sets permissions of files in the given package.

rpm --setugids PACKAGE_NAME

sets user/group ownership of files in the given package.

Ftp/Http Options

rpm can act as an FTP and/or HTTP client so that packages can be queried or installed from the internet. Package files for install, upgrade, and query operations may be specified as an **ftp** or **http** style URL:

`ftp://USER:PASSWORD@HOST:PORT/path/to/package.rpm`

If the **:PASSWORD** portion is omitted, the password will be prompted for (once per user/hostname pair). If both the user and password are omitted, anonymous **ftp** is used. In all cases, passive (PASV) **ftp** transfers are performed.

rpm allows the following options to be used with **ftp** URLs:

--ftpproxy **HOST**

The host **HOST** will be used as a proxy server for all **ftp** transfers, which allows users to **ftp** through firewall machines which use proxy systems. This option may also be specified by configuring the macro **%_ftpproxy**.

--ftpport **PORT**

The TCP **PORT** number to use for the **ftp** connection on the proxy **ftp** server instead of the default port. This option may also be specified by configuring the macro **%_ftpport**.

rpm allows the following options to be used with **http** URLs:

--httpproxy **HOST**

The host **HOST** will be used as a proxy server for all **http** transfers. This option may also be specified by configuring the macro **%_httpproxy**.

--httpport **PORT**

The TCP **PORT** number to use for the **http** connection on the proxy **http** server instead of the default port. This option may also be specified by configuring the macro **%_httpport**.

Legacy Issues

Executing **rpmbuild**

The build modes of **rpm** are now resident in the **/usr/bin/rpmbuild** executable. Install the package containing **rpmbuild** (usually **rpm-build**) and see ***rpmbuild*(8)** for documentation of all the **rpm** build modes.

Files

rpmrc Configuration

```
/usr/lib/rpm/rpmrc  
/usr/lib/rpm/redhat/rpmrc  
/etc/rpmrc  
~/.rpmrc
```

Macro Configuration

```
/usr/lib/rpm/macros
/usr/lib/rpm/redhat/macros
/etc/rpm/macros
~/.rpmmacros
```

Database

```
/var/lib/rpm/Basenames
/var/lib/rpm/Conflictname
/var/lib/rpm/Dirnames
/var/lib/rpm/Filemd5s
/var/lib/rpm/Group
/var/lib/rpm/Installtid
/var/lib/rpm/Name
/var/lib/rpm/Packages
/var/lib/rpm/Providename
/var/lib/rpm/Provideversion
/var/lib/rpm/Pubkeys
/var/lib/rpm/Removed
/var/lib/rpm/Requirename
/var/lib/rpm/Requireversion
/var/lib/rpm/Sha1header
/var/lib/rpm/Sigmd5
/var/lib/rpm/Triggername
```

Temporary

```
/var/tmp/rpm*
```

See Also

[popt\(3\)](#),
[rpm2cpio\(8\)](#),
[rpmbuild\(8\)](#),

rpm --help - as rpm supports customizing the options via popt aliases it's impossible to guarantee that what's described in the manual matches what's available.

<http://www.rpm.org/> <URL:<http://www.rpm.org/>>

Authors

Marc Ewing <marc@redhat.com>
Jeff Johnson <bjj@redhat.com>
Erik Troan <ewt@redhat.com>

Referenced By

[applydeltarpm\(8\)](#), [apt\(8\)](#), [apt.conf\(5\)](#), [autoupdate\(8\)](#), [cmake28\(1\)](#), [cmake28-gui\(1\)](#), [cmake28modules\(1\)](#), [compat_digest.plug\(1\)](#), [cpack28\(1\)](#), [demo.plug\(1\)](#), [demofiles.plug\(1\)](#), [febootstrap\(8\)](#), [file2pacdep.plug\(1\)](#), [nodoc.plug\(1\)](#), [rpm2paco\(8\)](#), [rpm4\(3\)](#), [rpm_selinux\(8\)](#), [rpmpcache\(8\)](#), [rpmconf\(8\)](#), [rpmdeps\(8\)](#), [rpmfile\(1\)](#), [rpmpeek\(1\)](#), [rpmreaper\(1\)](#), [rpmrebuild\(1\)](#), [rpmrebuild_plugins\(1\)](#), [set_tag.plug\(1\)](#), [superpaco\(8\)](#), [un_prelink.plug\(1\)](#), [uniq.plug\(1\)](#), [unset_tag.plug\(1\)](#), [yum-versionlock\(1\)](#)

NAME

rsync-ssl – a helper script for connecting to an ssl rsync daemon

SYNOPSIS

```
rsync-ssl [--type=SSL_TYPE] RSYNC_ARGS
```

DESCRIPTION

The rsync-ssl script helps you to run an rsync copy to/from an rsync daemon that requires ssl connections.

The script requires that you specify an rsync-daemon arg in the style of either **hostname::** (with 2 colons) or **rsync://hostname/**. The default port used for connecting is 874 (one higher than the normal 873) unless overridden in the environment. You can specify an overriding port via **--port** or by including it in the normal spot in the URL format, though both of those require your rsync version to be at least 3.2.0.

OPTIONS

If the **first** arg is a **--type=SSL_TYPE** option, the script will only use that particular program to open an ssl connection instead of trying to find an openssl or stunnel executable via a simple heuristic (assuming that the **RSYNC_SSL_TYPE** environment variable is not set as well -- see below). This option must specify one of **openssl** or **stunnel**. The equal sign is required for this particular option.

All the other options are passed through to the rsync command, so consult the **rsync(1)** manpage for more information on how it works.

ENVIRONMENT VARIABLES

The ssl helper scripts are affected by the following environment variables:

RSYNC_SSL_TYPE

Specifies the program type that should be used to open the ssl connection. It must be one of **openssl** or **stunnel**. The **--type=SSL_TYPE** option overrides this, when specified.

RSYNC_SSL_PORT

If specified, the value is the port number that is used as the default when the user does not specify a port in their rsync command. When not specified, the default port number is 874. (Note that older rsync versions (prior to 3.2.0) did not communicate an overriding port number value to the helper script.)

RSYNC_SSL_CERT

If specified, the value is a filename that contains a certificate to use for the connection.

RSYNC_SSL_CA_CERT

If specified, the value is a filename that contains a certificate authority certificate that is used to validate the connection.

RSYNC_SSL_OPENSSL

Specifies the openssl executable to run when the connection type is set to openssl. If unspecified, the \$PATH is searched for "openssl".

RSYNC_SSL_GNUTLS

Specifies the gnutls-cli executable to run when the connection type is set to gnutls. If unspecified, the \$PATH is searched for "gnutls-cli".

RSYNC_SSL_STUNNEL

Specifies the stunnel executable to run when the connection type is set to stunnel. If unspecified, the \$PATH is searched first for "stunnel4" and then for "stunnel".

EXAMPLES

```
rsync-ssl -av example.com::mod/ dest  
rsync-ssl --type=openssl -av example.com::mod/ dest  
rsync-ssl -av --port 9874 example.com::mod/ dest  
rsync-ssl -av rsync://example.com:9874/mod/ dest
```

SEE ALSO

rsync(1), rsyncd.conf(5)

CAVEATS

Note that using an stunnel connection requires at least version 4 of stunnel, which should be the case on modern systems. Also, it does not verify a connection against the CA certificate collection, so it only encrypts the connection without any cert validation unless you have specified the certificate environment options.

This script also supports a **--type=gnutls** option, but at the time of this release the gnutls-cli command was dropping output, making it unusable. If that bug has been fixed in your version, feel free to put gnutls into an exported RSYNC_SSL_TYPE environment variable to make its use the default.

BUGS

Please report bugs! See the web site at <https://rsync.samba.org/>.

VERSION

This man page is current for version 3.2.3 of rsync.

CREDITS

rsync is distributed under the GNU General Public License. See the file COPYING for details.

A web site is available at <https://rsync.samba.org/>. The site includes an FAQ-O-Matic which may cover questions unanswered by this manual page.

AUTHOR

This manpage was written by Wayne Davison.

Mailing lists for support and development are available at <https://lists.samba.org/>.

NAME

rsync – a fast, versatile, remote (and local) file-copying tool

SYNOPSIS

Local:

```
rsync [OPTION...] SRC... [DEST]
```

Access via remote shell:

Pull:

```
rsync [OPTION...] [USER@]HOST:SRC... [DEST]
```

Push:

```
rsync [OPTION...] SRC... [USER@]HOST:DEST
```

Access via rsync daemon:

Pull:

```
rsync [OPTION...] [USER@]HOST::SRC... [DEST]
```

```
rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
```

Push:

```
rsync [OPTION...] SRC... [USER@]HOST::DEST
```

```
rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

Usages with just one SRC arg and no DEST arg will list the source files instead of copying.

DESCRIPTION

Rsync is a fast and extraordinarily versatile file copying tool. It can copy locally, to/from another host over any remote shell, or to/from a remote rsync daemon. It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied. It is famous for its delta-transfer algorithm, which reduces the amount of data sent over the network by sending only the differences between the source files and the existing files in the destination. Rsync is widely used for backups and mirroring and as an improved copy command for everyday use.

Rsync finds files that need to be transferred using a "quick check" algorithm (by default) that looks for files that have changed in size or in last-modified time. Any changes in the other preserved attributes (as requested by options) are made on the destination file directly when the quick check indicates that the file's data does not need to be updated.

Some of the additional features of rsync are:

- o support for copying links, devices, owners, groups, and permissions
- o exclude and exclude-from options similar to GNU tar
- o a CVS exclude mode for ignoring the same files that CVS would ignore
- o can use any transparent remote shell, including ssh or rsh
- o does not require super-user privileges
- o pipelining of file transfers to minimize latency costs
- o support for anonymous or authenticated rsync daemons (ideal for mirroring)

GENERAL

Rsync copies files either to or from a remote host, or locally on the current host (it does not support copying files between two remote hosts).

There are two different ways for rsync to contact a remote system: using a remote-shell program as the transport (such as ssh or rsh) or contacting an rsync daemon directly via TCP. The remote-shell transport is used whenever the source or destination path contains a single colon (:) separator after a host specification. Contacting an rsync daemon directly happens when the source or destination path contains a double colon (::) separator after a host specification, OR when an rsync:// URL is specified (see also the "USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CONNECTION" section for an exception to this latter rule).

As a special case, if a single source arg is specified without a destination, the files are listed in an output format similar to "**ls -l**".

As expected, if neither the source or destination path specify a remote host, the copy occurs locally (see also the **--list-only** option).

Rsync refers to the local side as the client and the remote side as the server. Don't confuse server with an rsync daemon. A daemon is always a server, but a server can be either a daemon or a remote-shell spawned process.

SETUP

See the file README.md for installation instructions.

Once installed, you can use rsync to any machine that you can access via a remote shell (as well as some that you can access using the rsync daemon-mode protocol). For remote transfers, a modern rsync uses ssh for its communications, but it may have been configured to use a different remote shell by default, such as rsh or remsh.

You can also specify any remote shell you like, either by using the **-e** command line option, or by setting the RSYNC_RSH environment variable.

Note that rsync must be installed on both the source and destination machines.

USAGE

You use rsync in the same way you use rcp. You must specify a source and a destination, one of which may be remote.

Perhaps the best way to explain the syntax is with some examples:

```
rsync -t *.c foo:src/
```

This would transfer all files matching the pattern ***.c** from the current directory to the directory **src** on the machine **foo**. If any of the files already exist on the remote system then the rsync remote-update protocol is used to update the file by sending only the differences in the data. Note that the expansion of wildcards on the command-line (***.c**) into a list of files is handled by the shell before it runs rsync and not by rsync itself (exactly the same as all other Posix-style programs).

```
rsync -avz foo:src/bar /data/tmp
```

This would recursively transfer all files from the directory **src/bar** on the machine **foo** into the **/data/tmp/bar** directory on the local machine. The files are transferred in archive mode, which ensures that symbolic links, devices, attributes, permissions, ownerships, etc. are preserved in the transfer. Additionally, compression will be used to reduce the size of data portions of the transfer.

```
rsync -avz foo:src/bar/ /data/tmp
```

A trailing slash on the source changes this behavior to avoid creating an additional directory level at the destination. You can think of a trailing **/** on a source as meaning "copy the contents of this directory" as opposed to "copy the directory by name", but in both cases the attributes of the containing directory are transferred to the containing directory on the destination. In other words, each of the following commands copies the files in the same way, including their setting of the attributes of **/dest/foo**:

```
rsync -av /src/foo /dest
```

```
rsync -av /src/foo/ /dest/foo
```

Note also that host and module references don't require a trailing slash to copy the contents of the default directory. For example, both of these copy the remote directory's contents into **"/dest"**:

```
rsync -av host: /dest
```

```
rsync -av host::module /dest
```

You can also use rsync in local-only mode, where both the source and destination don't have a **:** in the name. In this case it behaves like an improved copy command.

Finally, you can list all the (listable) modules available from a particular rsync daemon by leaving off the module name:

```
rsync somehost.mydomain.com::
```

See the following section for more details.

ADVANCED USAGE

The syntax for requesting multiple files from a remote host is done by specifying additional remote-host args in the same style as the first, or with the hostname omitted. For instance, all these work:

```
rsync -av host:file1 :file2 host:file{3,4} /dest/  
rsync -av host::modname/file{1,2} host::modname/file3 /dest/  
rsync -av host::modname/file1 ::modname/file{3,4}
```

Older versions of rsync required using quoted spaces in the SRC, like these examples:

```
rsync -av host:'dir1/file1 dir2/file2' /dest  
rsync host::'modname/dir1/file1 modname/dir2/file2' /dest
```

This word-splitting still works (by default) in the latest rsync, but is not as easy to use as the first method.

If you need to transfer a filename that contains whitespace, you can either specify the **--protect-args** (-s) option, or you'll need to escape the whitespace in a way that the remote shell will understand. For instance:

```
rsync -av host:'file\ name\ with\ spaces' /dest
```

CONNECTING TO AN RSYNC DAEMON

It is also possible to use rsync without a remote shell as the transport. In this case you will directly connect to a remote rsync daemon, typically using TCP port 873. (This obviously requires the daemon to be running on the remote system, so refer to the STARTING AN RSYNC DAEMON TO ACCEPT CONNECTIONS section below for information on that.)

Using rsync in this way is the same as using it with a remote shell except that:

- o you either use a double colon :: instead of a single colon to separate the hostname from the path, or you use an rsync:// URL.
- o the first word of the "path" is actually a module name.
- o the remote daemon may print a message of the day when you connect.
- o if you specify no path name on the remote daemon then the list of accessible paths on the daemon will be shown.
- o if you specify no local destination then a listing of the specified files on the remote daemon is provided.
- o you must not specify the **--rsh** (-e) option (since that overrides the daemon connection to use ssh -- see USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CONNECTION below).

An example that copies all the files in a remote module named "src":

```
rsync -av host::src /dest
```

Some modules on the remote daemon may require authentication. If so, you will receive a password prompt when you connect. You can avoid the password prompt by setting the environment variable RSYNC_PASSWORD to the password you want to use or using the **--password-file** option. This may be useful when scripting rsync.

WARNING: On some systems environment variables are visible to all users. On those systems using **--password-file** is recommended.

You may establish the connection via a web proxy by setting the environment variable RSYNC_PROXY to a hostname:port pair pointing to your web proxy. Note that your web proxy's configuration must support proxy connections to port 873.

You may also establish a daemon connection using a program as a proxy by setting the environment variable RSYNC_CONNECT_PROG to the commands you wish to run in place of making a direct socket connection. The string may contain the escape "%H" to represent the hostname specified in the rsync

command (so use "%%" if you need a single "%" in your string). For example:

```
export RSYNC_CONNECT_PROG='ssh proxyhost nc %H 873'
rsync -av targethost1::module/src/ /dest/
rsync -av rsync://targethost2/module/src/ /dest/
```

The command specified above uses ssh to run nc (netcat) on a proxyhost, which forwards all data to port 873 (the rsync daemon) on the targethost (%H).

Note also that if the RSYNC_SHELL environment variable is set, that program will be used to run the RSYNC_CONNECT_PROG command instead of using the default shell of the `system()` call.

USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CONNECTION

It is sometimes useful to use various features of an rsync daemon (such as named modules) without actually allowing any new socket connections into a system (other than what is already required to allow remote-shell access). Rsync supports connecting to a host using a remote shell and then spawning a single-use "daemon" server that expects to read its config file in the home dir of the remote user. This can be useful if you want to encrypt a daemon-style transfer's data, but since the daemon is started up fresh by the remote user, you may not be able to use features such as chroot or change the uid used by the daemon. (For another way to encrypt a daemon transfer, consider using ssh to tunnel a local port to a remote machine and configure a normal rsync daemon on that remote host to only allow connections from "localhost".)

From the user's perspective, a daemon transfer via a remote-shell connection uses nearly the same command-line syntax as a normal rsync-daemon transfer, with the only exception being that you must explicitly set the remote shell program on the command-line with the `--rsh=COMMAND` option. (Setting the RSYNC_RSH in the environment will not turn on this functionality.) For example:

```
rsync -av --rsh=ssh host::module /dest
```

If you need to specify a different remote-shell user, keep in mind that the `user@` prefix in front of the host is specifying the rsync-user value (for a module that requires user-based authentication). This means that you must give the `-l user` option to ssh when specifying the remote-shell, as in this example that uses the short version of the `--rsh` option:

```
rsync -av -e "ssh -l ssh-user" rsync-user@host::module /dest
```

The "ssh-user" will be used at the ssh level; the "rsync-user" will be used to log-in to the "module".

STARTING AN RSYNC DAEMON TO ACCEPT CONNECTIONS

In order to connect to an rsync daemon, the remote system needs to have a daemon already running (or it needs to have configured something like inetd to spawn an rsync daemon for incoming connections on a particular port). For full information on how to start a daemon that will handle incoming socket connections, see the `rsyncd.conf(5)` man page --- that is the config file for the daemon, and it contains the full details for how to run the daemon (including stand-alone and inetd configurations).

If you're using one of the remote-shell transports for the transfer, there is no need to manually start an rsync daemon.

SORTED TRANSFER ORDER

Rsync always sorts the specified filenames into its internal transfer list. This handles the merging together of the contents of identically named directories, makes it easy to remove duplicate filenames, and may confuse someone when the files are transferred in a different order than what was given on the command-line.

If you need a particular file to be transferred prior to another, either separate the files into different rsync calls, or consider using `--delay-updates` (which doesn't affect the sorted transfer order, but does make the final file-updating phase happen much more rapidly).

EXAMPLES

Here are some examples of how I use rsync.

To backup my wife's home directory, which consists of large MS Word files and mail folders, I use a cron job that runs

```
rsync -Cavz . arvidsjaur:backup
```

each night over a PPP connection to a duplicate directory on my machine "arvidsjaur".

To synchronize my samba source trees I use the following Makefile targets:

```
get:
    rsync -avuzb --exclude '*~' samba:samba/
put:
    rsync -Cavuzb . samba:samba/
sync: get put
```

This allows me to sync with a CVS directory at the other end of the connection. I then do CVS operations on the remote machine, which saves a lot of time as the remote CVS protocol isn't very efficient.

I mirror a directory between my "old" and "new" ftp sites with the command:

```
rsync -az -e ssh --delete ~ftp/pub/samba nimbus:"~ftp/pub/tridge"
```

This is launched from cron every few hours.

OPTION SUMMARY

Here is a short summary of the options available in rsync. Please refer to the detailed description below for a complete description.

--verbose, -v	increase verbosity
--info=FLAGS	fine-grained informational verbosity
--debug=FLAGS	fine-grained debug verbosity
--stderr=e a c	change stderr output mode (default: errors)
--quiet, -q	suppress non-error messages
--no-motd	suppress daemon-mode MOTD
--checksum, -c	skip based on checksum, not mod-time & size
--archive, -a	archive mode is -rlptgoD (no -A,-X,-U,-N,-H)
--no-OPTION	turn off an implied OPTION (e.g. --no-D)
--recursive, -r	recurse into directories
--relative, -R	use relative path names
--no-implied-dirs	don't send implied dirs with --relative
--backup, -b	make backups (see --suffix & --backup-dir)
--backup-dir=DIR	make backups into hierarchy based in DIR
--suffix=SUFFIX	backup suffix (default ~ w/o --backup-dir)
--update, -u	skip files that are newer on the receiver
--inplace	update destination files in-place
--append	append data onto shorter files
--append-verify	--append w/old data in file checksum
--dirs, -d	transfer directories without recursing
--mkpath	create the destination's path component
--links, -l	copy symlinks as symlinks
--copy-links, -L	transform symlink into referent file/dir
--copy-unsafe-links	only "unsafe" symlinks are transformed
--safe-links	ignore symlinks that point outside the tree
--munge-links	munge symlinks to make them safe & unusable
--copy-dirlinks, -k	transform symlink to dir into referent dir
--keep-dirlinks, -K	treat symlinked dir on receiver as dir
--hard-links, -H	preserve hard links
--perms, -p	preserve permissions
--executability, -E	preserve executability
--chmod=CHMOD	affect file and/or directory permissions
--acls, -A	preserve ACLs (implies --perms)
--xattrs, -X	preserve extended attributes
--owner, -o	preserve owner (super-user only)
--group, -g	preserve group

```
--devices      preserve device files (super-user only)
--copy-devices    copy device contents as regular file
--specials      preserve special files
-D            same as --devices --specials
--times, -t      preserve modification times
--atimes, -U      preserve access (use) times
--open-noatime    avoid changing the atime on opened files
--crtimes, -N      preserve create times (newness)
--omit-dir-times, -O  omit directories from --times
--omit-link-times, -J  omit symlinks from --times
--super        receiver attempts super-user activities
--fake-super    store/recover privileged attrs using xattrs
--sparse, -S      turn sequences of nulls into sparse blocks
--preallocate    allocate dest files before writing them
--write-devices   write to devices as files (implies --inplace)
--dry-run, -n      perform a trial run with no changes made
--whole-file, -W      copy files whole (w/o delta-xfer algorithm)
--checksum-choice=STR  choose the checksum algorithm (aka --cc)
--one-file-system, -x  don't cross filesystem boundaries
--block-size=SIZE, -B  force a fixed checksum block-size
--rsh=COMMAND, -e      specify the remote shell to use
--rsync-path=PROGRAM   specify the rsync to run on remote machine
--existing      skip creating new files on receiver
--ignore-existing    skip updating files that exist on receiver
--remove-source-files  sender removes synchronized files (non-dir)
--del          an alias for --delete-during
--delete        delete extraneous files from dest dirs
--delete-before    receiver deletes before xfer, not during
--delete-during    receiver deletes during the transfer
--delete-delay     find deletions during, delete after
--delete-after     receiver deletes after transfer, not during
--delete-excluded   also delete excluded files from dest dirs
--ignore-missing-args  ignore missing source args without error
--delete-missing-args  delete missing source args from destination
--ignore-errors     delete even if there are I/O errors
--force          force deletion of dirs even if not empty
--max-delete=NUM      don't delete more than NUM files
--max-size=SIZE      don't transfer any file larger than SIZE
--min-size=SIZE      don't transfer any file smaller than SIZE
--max-alloc=SIZE      change a limit relating to memory alloc
--partial        keep partially transferred files
--partial-dir=DIR     put a partially transferred file into DIR
--delay-updates    put all updated files into place at end
--prune-empty-dirs, -m  prune empty directory chains from file-list
--numeric-ids      don't map uid/gid values by user/group name
--usermap=STRING     custom username mapping
--groupmap=STRING    custom groupname mapping
--chown=USER:GROUP    simple username/groupname mapping
--timeout=SECONDS    set I/O timeout in seconds
--contimeout=SECONDS   set daemon connection timeout in seconds
--ignore-times, -I      don't skip files that match size and time
--size-only       skip files that match in size
--modify-window=NUM, -@  set the accuracy for mod-time comparisons
--temp-dir=DIR, -T      create temporary files in directory DIR
```

```
--fuzzy, -y      find similar file for basis if no dest file
--compare-dest=DIR  also compare destination files relative to DIR
--copy-dest=DIR    ... and include copies of unchanged files
--link-dest=DIR    hardlink to files in DIR when unchanged
--compress, -z     compress file data during the transfer
--compress-choice=STR choose the compression algorithm (aka --zc)
--compress-level=NUM explicitly set compression level (aka --zl)
--skip-compress=LIST skip compressing files with suffix in LIST
--cvs-exclude, -C auto-ignore files in the same way CVS does
--filter=RULE, -f add a file-filtering RULE
-F                 same as --filter='dir-merge ./rsync-filter'
                   repeated: --filter='-' .rsync-filter'
--exclude=PATTERN  exclude files matching PATTERN
--exclude-from=FILE read exclude patterns from FILE
--include=PATTERN   don't exclude files matching PATTERN
--include-from=FILE read include patterns from FILE
--files-from=FILE   read list of source-file names from FILE
--from0, -0         all *-from/filter files are delimited by 0s
--protect-args, -s  no space-splitting; wildcard chars only
--copy-as=USER[:GROUP] specify user & optional group for the copy
--address=ADDRESS   bind address for outgoing socket to daemon
--port=PORT         specify double-colon alternate port number
--sockopt=OPTIONS   specify custom TCP options
--blocking-io       use blocking I/O for the remote shell
--outbuf=N|L|B      set out buffering to None, Line, or Block
--stats              give some file-transfer stats
--8-bit-output, -8   leave high-bit chars unescaped in output
--human-readable, -h output numbers in a human-readable format
--progress           show progress during transfer
-P                 same as --partial --progress
--itemize-changes, -i output a change-summary for all updates
--remote-option=OPT, -M send OPTION to the remote side only
--out-format=FORMAT  output updates using the specified FORMAT
--log-file=FILE      log what we're doing to the specified FILE
--log-file-format=FMT log updates using the specified FMT
--password-file=FILE read daemon-access password from FILE
--early-input=FILE   use FILE for daemon's early exec input
--list-only          list the files instead of copying them
--bwlimit=RATE        limit socket I/O bandwidth
--stop-after=MINS    Stop rsync after MINS minutes have elapsed
--stop-at=y-m-dTh:m  Stop rsync at the specified point in time
--write-batch=FILE   write a batched update to FILE
--only-write-batch=FILE like --write-batch but w/o updating dest
--read-batch=FILE    read a batched update from FILE
--protocol=NUM        force an older protocol version to be used
--iconv=CONVERT_SPEC request charset conversion of filenames
--checksum-seed=NUM  set block/file checksum seed (advanced)
--ipv4, -4            prefer IPv4
--ipv6, -6            prefer IPv6
--version, -V          print the version + other info and exit
--help, -h (*)        show this help (* -h is help only on its own)
```

Rsync can also be run as a daemon, in which case the following options are accepted:

```
--daemon           run as an rsync daemon
--address=ADDRESS  bind to the specified address
```

```
--bwlimit=RATE      limit socket I/O bandwidth
--config=FILE       specify alternate rsyncd.conf file
--dparam=OVERRIDE, -M  override global daemon config parameter
--no-detach        do not detach from the parent
--port=PORT         listen on alternate port number
--log-file=FILE     override the "log file" setting
--log-file-format=FMT  override the "log format" setting
--sockopts=OPTIONS  specify custom TCP options
--verbose, -v       increase verbosity
--ipv4, -4          prefer IPv4
--ipv6, -6          prefer IPv6
--help, -h          show this help (when used with --daemon)
```

OPTIONS

Rsync accepts both long (double-dash + word) and short (single-dash + letter) options. The full list of the available options are described below. If an option can be specified in more than one way, the choices are comma-separated. Some options only have a long variant, not a short. If the option takes a parameter, the parameter is only listed after the long variant, even though it must also be specified for the short. When specifying a parameter, you can either use the form **--option=param** or replace the '=' with whitespace. The parameter may need to be quoted in some manner for it to survive the shell's command-line parsing. Keep in mind that a leading tilde (~) in a filename is substituted by your shell, so **--option=~foo** will not change the tilde into your home directory (remove the '=' for that).

--help, -h (*)

Print a short help page describing the options available in rsync and exit. (*) The **-h** short option will only invoke **--help** when used without other options since it normally means **--human-readable**.

--version, -V

Print the rsync version plus other info and exit.

The output includes the default list of checksum algorithms, the default list of compression algorithms, a list of compiled-in capabilities, a link to the rsync web site, and some license/copyright info.

--verbose, -v

This option increases the amount of information you are given during the transfer. By default, rsync works silently. A single **-v** will give you information about what files are being transferred and a brief summary at the end. Two **-v** options will give you information on what files are being skipped and slightly more information at the end. More than two **-v** options should only be used if you are debugging rsync.

In a modern rsync, the **-v** option is equivalent to the setting of groups of **--info** and **--debug** options. You can choose to use these newer options in addition to, or in place of using **--verbose**, as any fine-grained settings override the implied settings of **-v**. Both **--info** and **--debug** have a way to ask for help that tells you exactly what flags are set for each increase in verbosity.

However, do keep in mind that a daemon's "**max verbosity**" setting will limit how high of a level the various individual flags can be set on the daemon side. For instance, if the max is 2, then any info and/or debug flag that is set to a higher value than what would be set by **-vv** will be downgraded to the **-vv** level in the daemon's logging.

--info=FLAGS

This option lets you have fine-grained control over the information output you want to see. An individual flag name may be followed by a level number, with 0 meaning to silence that output, 1 being the default output level, and higher numbers increasing the output of that flag (for those that support higher levels). Use **--info=help** to see all the available flag names, what they output, and what flag names are added for each increase in the verbose level. Some examples:

```
rsync -a --info=progress2 src/ dest/
rsync -avv --info=stats2,misc1,flist0 src/ dest/
```

Note that **--info=name**'s output is affected by the **--out-format** and **--itemize-changes (-i)** options. See those options for more information on what is output and when.

This option was added to 3.1.0, so an older rsync on the server side might reject your attempts at fine-grained control (if one or more flags needed to be send to the server and the server was too old to understand them). See also the "**max verbosity**" caveat above when dealing with a daemon.

--debug=FLAGS

This option lets you have fine-grained control over the debug output you want to see. An individual flag name may be followed by a level number, with 0 meaning to silence that output, 1 being the default output level, and higher numbers increasing the output of that flag (for those that support higher levels). Use **--deb ug=help** to see all the available flag names, what they output, and what flag names are added for each increase in the verbose level. Some examples:

```
rsync -avvv --debug=none src/ dest/
rsync -avA --del --debug=del2,acl src/ dest/
```

Note that some debug messages will only be output when **--stderr=all** is specified, especially those pertaining to I/O and buffer debugging.

Beginning in 3.2.0, this option is no longer auto-forwarded to the server side in order to allow you to specify different debug values for each side of the transfer, as well as to specify a new debug option that is only present in one of the rsync versions. If you want to duplicate the same option on both sides, using brace expansion is an easy way to save you some typing. This works in zsh and bash:

```
rsync -av { -M, }--debug=del2 src/ dest/
```

--stderr=errors|all|client

This option controls which processes output to stderr and if info messages are also changed to stderr. The mode strings can be abbreviated, so feel free to use a single letter value. The 3 possible choices are:

- o **errors** – (the default) causes all the rsync processes to send an error directly to stderr, even if the process is on the remote side of the transfer. Info messages are sent to the client side via the protocol stream. If stderr is not available (i.e. when directly connecting with a daemon via a socket) errors fall back to being sent via the protocol stream.
- o **all** – causes all rsync messages (info and error) to get written directly to stderr from all (possible) processes. This causes stderr to become line-buffered (instead of raw) and eliminates the ability to divide up the info and error messages by file handle. For those doing debugging or using several levels of verbosity, this option can help to avoid clogging up the transfer stream (which should prevent any chance of a deadlock bug hanging things up). It also allows **--debug** to enable some extra I/O related messages.
- o **client** – causes all rsync messages to be sent to the client side via the protocol stream. One client process outputs all messages, with errors on stderr and info messages on std-out. This was the default in older rsync versions, but can cause error delays when a lot of transfer data is ahead of the messages. If you're pushing files to an older rsync, you may want to use **--stderr=all** since that idiom has been around for several releases.

This option was added in rsync 3.2.3. This version also began the forwarding of a non-default setting to the remote side, though rsync uses the backward-compatible options **--msgs2stderr** and **--no-msgs2stderr** to represent the **all** and **client** settings, respectively. A newer rsync will continue to accept these older option names to maintain compatibility.

--quiet, -q

This option decreases the amount of information you are given during the transfer, notably suppressing information messages from the remote server. This option is useful when invoking rsync

from cron.

--no-motd

This option affects the information that is output by the client at the start of a daemon transfer. This suppresses the message-of-the-day (MOTD) text, but it also affects the list of modules that the daemon sends in response to the "rsync host::" request (due to a limitation in the rsync protocol), so omit this option if you want to request the list of modules from the daemon.

--ignore-times, -I

Normally rsync will skip any files that are already the same size and have the same modification timestamp. This option turns off this "quick check" behavior, causing all files to be updated.

--size-only

This modifies rsync's "quick check" algorithm for finding files that need to be transferred, changing it from the default of transferring files with either a changed size or a changed last-modified time to just looking for files that have changed in size. This is useful when starting to use rsync after using another mirroring system which may not preserve timestamps exactly.

--modify-window=NUM, -@

When comparing two timestamps, rsync treats the timestamps as being equal if they differ by no more than the modify-window value. The default is 0, which matches just integer seconds. If you specify a negative value (and the receiver is at least version 3.1.3) then nanoseconds will also be taken into account. Specifying 1 is useful for copies to/from MS Windows FAT filesystems, because FAT represents times with a 2-second resolution (allowing times to differ from the original by up to 1 second).

If you want all your transfers to default to comparing nanoseconds, you can create a `~/.popt` file and put these lines in it:

```
rsync alias -a -a@-1
rsync alias -t -t@-1
```

With that as the default, you'd need to specify **--modify-window=0** (aka **-@0**) to override it and ignore nanoseconds, e.g. if you're copying between ext3 and ext4, or if the receiving rsync is older than 3.1.3.

--checksum, -c

This changes the way rsync checks if the files have been changed and are in need of a transfer. Without this option, rsync uses a "quick check" that (by default) checks if each file's size and time of last modification match between the sender and receiver. This option changes this to compare a 128-bit checksum for each file that has a matching size. Generating the checksums means that both sides will expend a lot of disk I/O reading all the data in the files in the transfer, so this can slow things down significantly (and this is prior to any reading that will be done to transfer changed files)

The sending side generates its checksums while it is doing the file-system scan that builds the list of the available files. The receiver generates its checksums when it is scanning for changed files, and will checksum any file that has the same size as the corresponding sender's file: files with either a changed size or a changed checksum are selected for transfer.

Note that rsync always verifies that each *transferred* file was correctly reconstructed on the receiving side by checking a whole-file checksum that is generated as the file is transferred, but that automatic after-the-transfer verification has nothing to do with this option's before-the-transfer "Does this file need to be updated?" check.

The checksum used is auto-negotiated between the client and the server, but can be overridden using either the **--checksum-choice** (**--cc**) option or an environment variable that is discussed in that option's section.

--archive, -a

This is equivalent to **-rlptgoD**. It is a quick way of saying you want recursion and want to preserve almost everything. Be aware that it does **not** include preserving ACLs (**-A**), xattrs (**-X**),

atimes (**-U**), ctimes (**-N**), nor the finding and preserving of hardlinks (**-H**).

The only exception to the above equivalence is when **--files-from** is specified, in which case **-r** is not implied.

--no-OPTION

You may turn off one or more implied options by prefixing the option name with "no-". Not all options may be prefixed with a "no-": only options that are implied by other options (e.g. **--no-D**, **--no-perms**) or have different defaults in various circumstances (e.g. **--no-whole-file**, **--no-blocking-io**, **--no-dirs**). You may specify either the short or the long option name after the "no-" prefix (e.g. **--no-R** is the same as **--no-relative**).

For example: if you want to use **-a** (**--archive**) but don't want **-o** (**--owner**), instead of converting **-a** into **-rlptgD**, you could specify **-a --no-o** (or **-a --no-owner**).

The order of the options is important: if you specify **--no-r -a**, the **-r** option would end up being turned on, the opposite of **-a --no-r**. Note also that the side-effects of the **--files-from** option are NOT positional, as it affects the default state of several options and slightly changes the meaning of **-a** (see the **--files-from** option for more details).

--recursive, -r

This tells rsync to copy directories recursively. See also **--dirs (-d)**.

Beginning with rsync 3.0.0, the recursive algorithm used is now an incremental scan that uses much less memory than before and begins the transfer after the scanning of the first few directories have been completed. This incremental scan only affects our recursion algorithm, and does not change a non-recursive transfer. It is also only possible when both ends of the transfer are at least version 3.0.0.

Some options require rsync to know the full file list, so these options disable the incremental recursion mode. These include: **--delete-before**, **--delete-after**, **--prune-empty-dirs**, and **--delay-updates**. Because of this, the default delete mode when you specify **--delete** is now **--delete-during** when both ends of the connection are at least 3.0.0 (use **--del** or **--delete-during** to request this improved deletion mode explicitly). See also the **--delete-delay** option that is a better choice than using **--delete-after**.

Incremental recursion can be disabled using the **--no-inc-recursive** option or its shorter **--no-i-r** alias.

--relative, -R

Use relative paths. This means that the full path names specified on the command line are sent to the server rather than just the last parts of the filenames. This is particularly useful when you want to send several different directories at the same time. For example, if you used this command:

```
rsync -av /foo/bar/baz.c remote:/tmp/
```

would create a file named *baz.c* in */tmp/* on the remote machine. If instead you used

```
rsync -avR /foo/bar/baz.c remote:/tmp/
```

then a file named */tmp/foo/bar/baz.c* would be created on the remote machine, preserving its full path. These extra path elements are called "implied directories" (i.e. the "foo" and the "foo/bar" directories in the above example).

Beginning with rsync 3.0.0, rsync always sends these implied directories as real directories in the file list, even if a path element is really a symlink on the sending side. This prevents some really unexpected behaviors when copying the full path of a file that you didn't realize had a symlink in its path. If you want to duplicate a server-side symlink, include both the symlink via its path, and referent directory via its real path. If you're dealing with an older rsync on the sending side, you may need to use the **--no-implied-dirs** option.

It is also possible to limit the amount of path information that is sent as implied directories for each path you specify. With a modern rsync on the sending side (beginning with 2.6.7), you can

insert a dot and a slash into the source path, like this:

```
rsync -avR /foo./bar/baz.c remote:/tmp/
```

That would create /tmp/bar/baz.c on the remote machine. (Note that the dot must be followed by a slash, so "/foo./" would not be abbreviated.) For older rsync versions, you would need to use a chdir to limit the source path. For example, when pushing files:

```
(cd /foo; rsync -avR bar/baz.c remote:/tmp/)
```

(Note that the parens put the two commands into a sub-shell, so that the "cd" command doesn't remain in effect for future commands.) If you're pulling files from an older rsync, use this idiom (but only for a non-daemon transfer):

```
rsync -avR --rsync-path="cd /foo; rsync" \
      remote:bar/baz.c /tmp/
```

--no-implied-dirs

This option affects the default behavior of the **--relative** option. When it is specified, the attributes of the implied directories from the source names are not included in the transfer. This means that the corresponding path elements on the destination system are left unchanged if they exist, and any missing implied directories are created with default attributes. This even allows these implied path elements to have big differences, such as being a symlink to a directory on the receiving side.

For instance, if a command-line arg or a files-from entry told rsync to transfer the file "path/foo/file", the directories "path" and "path/foo" are implied when **--relative** is used. If "path/foo" is a symlink to "bar" on the destination system, the receiving rsync would ordinarily delete "path/foo", recreate it as a directory, and receive the file into the new directory. With **--no-implied-dirs**, the receiving rsync updates "path/foo/file" using the existing path elements, which means that the file ends up being created in "path/bar". Another way to accomplish this link preservation is to use the **--keep-dirlinks** option (which will also affect symlinks to directories in the rest of the transfer).

When pulling files from an rsync older than 3.0.0, you may need to use this option if the sending side has a symlink in the path you request and you wish the implied directories to be transferred as normal directories.

--backup, -b

With this option, preexisting destination files are renamed as each file is transferred or deleted. You can control where the backup file goes and what (if any) suffix gets appended using the **--backup-dir** and **--suffix** options.

Note that if you don't specify **--backup-dir**, (1) the **--omit-dir-times** option will be forced on, and (2) if **--delete** is also in effect (without **--delete-excluded**), rsync will add a "protect" filter-rule for the backup suffix to the end of all your existing excludes (e.g. **-f "P *~"**). This will prevent previously backed-up files from being deleted. Note that if you are supplying your own filter rules, you may need to manually insert your own exclude/protect rule somewhere higher up in the list so that it has a high enough priority to be effective (e.g., if your rules specify a trailing inclusion/exclusion of *, the auto-added rule would never be reached).

--backup-dir=DIR

This implies the **--backup** option, and tells rsync to store all backups in the specified directory on the receiving side. This can be used for incremental backups. You can additionally specify a backup suffix using the **--suffix** option (otherwise the files backed up in the specified directory will keep their original filenames).

Note that if you specify a relative path, the backup directory will be relative to the destination directory, so you probably want to specify either an absolute path or a path that starts with "../". If an rsync daemon is the receiver, the backup dir cannot go outside the module's path hierarchy, so take extra care not to delete it or copy into it.

--suffix=SUFFIX

This option allows you to override the default backup suffix used with the **--backup (-b)** option. The default suffix is a ~ if no **--backup-dir** was specified, otherwise it is an empty string.

--update, -u

This forces rsync to skip any files which exist on the destination and have a modified time that is newer than the source file. (If an existing destination file has a modification time equal to the source file's, it will be updated if the sizes are different.)

Note that this does not affect the copying of dirs, symlinks, or other special files. Also, a difference of file format between the sender and receiver is always considered to be important enough for an update, no matter what date is on the objects. In other words, if the source has a directory where the destination has a file, the transfer would occur regardless of the timestamps.

This option is a transfer rule, not an exclude, so it doesn't affect the data that goes into the file-lists, and thus it doesn't affect deletions. It just limits the files that the receiver requests to be transferred.

--inplace

This option changes how rsync transfers a file when its data needs to be updated: instead of the default method of creating a new copy of the file and moving it into place when it is complete, rsync instead writes the updated data directly to the destination file.

This has several effects:

- o Hard links are not broken. This means the new data will be visible through other hard links to the destination file. Moreover, attempts to copy differing source files onto a multiply-linked destination file will result in a "tug of war" with the destination data changing back and forth.
- o In-use binaries cannot be updated (either the OS will prevent this from happening, or binaries that attempt to swap-in their data will misbehave or crash).
- o The file's data will be in an inconsistent state during the transfer and will be left that way if the transfer is interrupted or if an update fails.
- o A file that rsync cannot write to cannot be updated. While a super user can update any file, a normal user needs to be granted write permission for the open of the file for writing to be successful.
- o The efficiency of rsync's delta-transfer algorithm may be reduced if some data in the destination file is overwritten before it can be copied to a position later in the file. This does not apply if you use **--backup**, since rsync is smart enough to use the backup file as the basis file for the transfer.

WARNING: you should not use this option to update files that are being accessed by others, so be careful when choosing to use this for a copy.

This option is useful for transferring large files with block-based changes or appended data, and also on systems that are disk bound, not network bound. It can also help keep a copy-on-write filesystem snapshot from diverging the entire contents of a file that only has minor changes.

The option implies **--partial** (since an interrupted transfer does not delete the file), but conflicts with **--partial-dir** and **--delay-updates**. Prior to rsync 2.6.4 **--inplace** was also incompatible with **--compare-dest** and **--link-dest**.

--append

This special copy mode only works to efficiently update files that are known to be growing larger where any existing content on the receiving side is also known to be the same as the content on the sender. The use of **--append** can be **dangerous** if you aren't 100% sure that all the files in the transfer are shared, growing files. You should thus use filter rules to ensure that you weed out any files that do not fit this criteria.

Rsync updates these growing file in-place without verifying any of the existing content in the file (it only verifies the content that it is appending). Rsync skips any files that exist on the receiving side that are not shorter than the associated file on the sending side (which means that new files are transferred).

This does not interfere with the updating of a file's non-content attributes (e.g. permissions, ownership, etc.) when the file does not need to be transferred, nor does it affect the updating of any directories or non-regular files.

--append-verify

This special copy mode works like **--append** except that all the data in the file is included in the checksum verification (making it much less efficient but also potentially safer). This option **can be dangerous** if you aren't 100% sure that all the files in the transfer are shared, growing files. See the **--append** option for more details.

Note: prior to rsync 3.0.0, the **--append** option worked like **--append-verify**, so if you are interacting with an older rsync (or the transfer is using a protocol prior to 30), specifying either append option will initiate an **--append-verify** transfer.

--dirs, -d

Tell the sending side to include any directories that are encountered. Unlike **--recursive**, a directory's contents are not copied unless the directory name specified is `".` or ends with a trailing slash (e.g. `..`, `"dir/."`, `"dir/"`, etc.). Without this option or the **--recursive** option, rsync will skip all directories it encounters (and output a message to that effect for each one). If you specify both **--dirs** and **--recursive**, **--recursive** takes precedence.

The **--dirs** option is implied by the **--files-from** option or the **--list-only** option (including an implied **--list-only** usage) if **--recursive** wasn't specified (so that directories are seen in the listing). Specify **--no-dirs** (or **--no-d**) if you want to turn this off.

There is also a backward-compatibility helper option, **--old-dirs** (or **--old-d**) that tells rsync to use a hack of **-r --exclude='/*/*'** to get an older rsync to list a single directory without recursing.

--mkpath

Create a missing path component of the destination arg. This allows rsync to create multiple levels of missing destination dirs and to create a path in which to put a single renamed file. Keep in mind that you'll need to supply a trailing slash if you want the entire destination path to be treated as a directory when copying a single arg (making rsync behave the same way that it would if the path component of the destination had already existed).

For example, the following creates a copy of file `foo` as `bar` in the `sub/dir` directory, creating dirs `"sub"` and `"sub/dir"` if either do not yet exist:

```
rsync -ai --mkpath foo sub/dir/bar
```

If you instead ran the following, it would have created file `foo` in the `sub/dir/bar` directory:

```
rsync -ai --mkpath foo sub/dir/bar/
```

--links, -l

When symlinks are encountered, recreate the symlink on the destination.

--copy-links, -L

When symlinks are encountered, the item that they point to (the referent) is copied, rather than the symlink. In older versions of rsync, this option also had the side-effect of telling the receiving side to follow symlinks, such as symlinks to directories. In a modern rsync such as this one, you'll need to specify **--keep-dirlinks (-K)** to get this extra behavior. The only exception is when sending files to an rsync that is too old to understand **-K** — in that case, the **-L** option will still have the side-effect of **-K** on that older receiving rsync.

--copy-unsafe-links

This tells rsync to copy the referent of symbolic links that point outside the copied tree. Absolute symlinks are also treated like ordinary files, and so are any symlinks in the source path itself when

--relative is used. This option has no additional effect if **--copy-links** was also specified.

Note that the cut-off point is the top of the transfer, which is the part of the path that rsync isn't mentioning in the verbose output. If you copy "/src/subdir" to "/dest/" then the "subdir" directory is a name inside the transfer tree, not the top of the transfer (which is /src) so it is legal for created relative symlinks to refer to other names inside the /src and /dest directories. If you instead copy "/src/subdir/" (with a trailing slash) to "/dest/subdir" that would not allow symlinks to any files outside of "subdir".

--safe-links

This tells rsync to ignore any symbolic links which point outside the copied tree. All absolute symlinks are also ignored. Using this option in conjunction with **--relative** may give unexpected results.

--munge-links

This option tells rsync to (1) modify all symlinks on the receiving side in a way that makes them unusable but recoverable (see below), or (2) to unmunge symlinks on the sending side that had been stored in a munged state. This is useful if you don't quite trust the source of the data to not try to slip in a symlink to a unexpected place.

The way rsync disables the use of symlinks is to prefix each one with the string "/rsyncd-munged/". This prevents the links from being used as long as that directory does not exist. When this option is enabled, rsync will refuse to run if that path is a directory or a symlink to a directory.

The option only affects the client side of the transfer, so if you need it to affect the server, specify it via **--remote-option**. (Note that in a local transfer, the client side is the sender.)

This option has no affect on a daemon, since the daemon configures whether it wants munged symlinks via its "**munge symlinks**" parameter. See also the "munge-symlinks" perl script in the support directory of the source code.

--copy-dirlinks, -k

This option causes the sending side to treat a symlink to a directory as though it were a real directory. This is useful if you don't want symlinks to non-directories to be affected, as they would be using **--copy-links**.

Without this option, if the sending side has replaced a directory with a symlink to a directory, the receiving side will delete anything that is in the way of the new symlink, including a directory hierarchy (as long as **--force** or **--delete** is in effect).

See also **--keep-dirlinks** for an analogous option for the receiving side.

--copy-dirlinks applies to all symlinks to directories in the source. If you want to follow only a few specified symlinks, a trick you can use is to pass them as additional source args with a trailing slash, using **--relative** to make the paths match up right. For example:

```
rsync -r --relative src/./ src./follow-me/ dest/
```

This works because rsync calls **Istat(2)** on the source arg as given, and the trailing slash makes **Istat(2)** follow the symlink, giving rise to a directory in the file-list which overrides the symlink found during the scan of "src./".

--keep-dirlinks, -K

This option causes the receiving side to treat a symlink to a directory as though it were a real directory, but only if it matches a real directory from the sender. Without this option, the receiver's symlink would be deleted and replaced with a real directory.

For example, suppose you transfer a directory "foo" that contains a file "file", but "foo" is a symlink to directory "bar" on the receiver. Without **--keep-dirlinks**, the receiver deletes symlink "foo", recreates it as a directory, and receives the file into the new directory. With **--keep-dirlinks**, the receiver keeps the symlink and "file" ends up in "bar".

One note of caution: if you use **--keep-dirlinks**, you must trust all the symlinks in the copy! If it is possible for an untrusted user to create their own symlink to any directory, the user could then (on a subsequent copy) replace the symlink with a real directory and affect the content of whatever directory the symlink references. For backup copies, you are better off using something like a bind mount instead of a symlink to modify your receiving hierarchy.

See also **--copy-dirlinks** for an analogous option for the sending side.

--hard-links, -H

This tells rsync to look for hard-linked files in the source and link together the corresponding files on the destination. Without this option, hard-linked files in the source are treated as though they were separate files.

This option does NOT necessarily ensure that the pattern of hard links on the destination exactly matches that on the source. Cases in which the destination may end up with extra hard links include the following:

- o If the destination contains extraneous hard-links (more linking than what is present in the source file list), the copying algorithm will not break them explicitly. However, if one or more of the paths have content differences, the normal file-update process will break those extra links (unless you are using the **--inplace** option).
- o If you specify a **--link-dest** directory that contains hard links, the linking of the destination files against the **--link-dest** files can cause some paths in the destination to become linked together due to the **--link-dest** associations.

Note that rsync can only detect hard links between files that are inside the transfer set. If rsync updates a file that has extra hard-link connections to files outside the transfer, that linkage will be broken. If you are tempted to use the **--inplace** option to avoid this breakage, be very careful that you know how your files are being updated so that you are certain that no unintended changes happen due to lingering hard links (and see the **--inplace** option for more caveats).

If incremental recursion is active (see **--recursive**), rsync may transfer a missing hard-linked file before it finds that another link for that contents exists elsewhere in the hierarchy. This does not affect the accuracy of the transfer (i.e. which files are hard-linked together), just its efficiency (i.e. copying the data for a new, early copy of a hard-linked file that could have been found later in the transfer in another member of the hard-linked set of files). One way to avoid this inefficiency is to disable incremental recursion using the **--no-inc-recursive** option.

--perms, -p

This option causes the receiving rsync to set the destination permissions to be the same as the source permissions. (See also the **--chmod** option for a way to modify what rsync considers to be the source permissions.)

When this option is *off*, permissions are set as follows:

- o Existing files (including updated files) retain their existing permissions, though the **--executability** option might change just the execute permission for the file.
- o New files get their "normal" permission bits set to the source file's permissions masked with the receiving directory's default permissions (either the receiving process's umask, or the permissions specified via the destination directory's default ACL), and their special permission bits disabled except in the case where a new directory inherits a setgid bit from its parent directory.

Thus, when **--perms** and **--executability** are both disabled, rsync's behavior is the same as that of other file-copy utilities, such as **cp(1)** and **tar(1)**.

In summary: to give destination files (both old and new) the source permissions, use **--perms**. To give new files the destination-default permissions (while leaving existing files unchanged), make sure that the **--perms** option is off and use **--chmod=ugo=rwX** (which ensures that all non-masked bits get enabled). If you'd care to make this latter behavior easier to type, you could

define a popt alias for it, such as putting this line in the file `~/.popt` (the following defines the `-Z` option, and includes `--no-g` to use the default group of the destination dir):

```
rsync alias -Z --no-p --no-g --chmod=ugo=rwX
```

You could then use this new option in a command such as this one:

```
rsync -avZ src/ dest/
```

(Caveat: make sure that `-a` does not follow `-Z`, or it will re-enable the two `--no-*` options mentioned above.)

The preservation of the destination's setgid bit on newly-created directories when `--perms` is off was added in rsync 2.6.7. Older rsync versions erroneously preserved the three special permission bits for newly-created files when `--perms` was off, while overriding the destination's setgid bit setting on a newly-created directory. Default ACL observance was added to the ACL patch for rsync 2.6.7, so older (or non-ACL-enabled) rsyncs use the umask even if default ACLs are present. (Keep in mind that it is the version of the receiving rsync that affects these behaviors.)

--executability, -E

This option causes rsync to preserve the executability (or non-executability) of regular files when `--perms` is not enabled. A regular file is considered to be executable if at least one 'x' is turned on in its permissions. When an existing destination file's executability differs from that of the corresponding source file, rsync modifies the destination file's permissions as follows:

- o To make a file non-executable, rsync turns off all its 'x' permissions.
- o To make a file executable, rsync turns on each 'x' permission that has a corresponding 'r' permission enabled.

If `--perms` is enabled, this option is ignored.

--acls, -A

This option causes rsync to update the destination ACLs to be the same as the source ACLs. The option also implies `--perms`.

The source and destination systems must have compatible ACL entries for this option to work properly. See the `--fak e-super` option for a way to backup and restore ACLs that are not compatible.

--xattrs, -X

This option causes rsync to update the destination extended attributes to be the same as the source ones.

For systems that support extended-attribute namespaces, a copy being done by a super-user copies all namespaces except `system.*`. A normal user only copies the `user.*` namespace. To be able to backup and restore non-user namespaces as a normal user, see the `--fake-super` option.

The above name filtering can be overridden by using one or more filter options with the `x` modifier. When you specify an xattr-affecting filter rule, rsync requires that you do your own system/user filtering, as well as any additional filtering for what xattr names are copied and what names are allowed to be deleted. For example, to skip the system namespace, you could specify:

```
--filter=' -x system.*'
```

To skip all namespaces except the user namespace, you could specify a negated-user match:

```
--filter=' -x! user.*'
```

To prevent any attributes from being deleted, you could specify a receiver-only rule that excludes all names:

```
--filter=' -xr *'
```

Note that the `-X` option does not copy rsync's special xattr values (e.g. those used by `--fake-super`) unless you repeat the option (e.g. `-XX`). This "copy all xattrs" mode cannot be used with

--fake-super.**--chmod=CHMOD**

This option tells rsync to apply one or more comma-separated "chmod" modes to the permission of the files in the transfer. The resulting value is treated as though it were the permissions that the sending side supplied for the file, which means that this option can seem to have no effect on existing files if **--perms** is not enabled.

In addition to the normal parsing rules specified in the **chmod(1)** manpage, you can specify an item that should only apply to a directory by prefixing it with a 'D', or specify an item that should only apply to a file by prefixing it with a 'F'. For example, the following will ensure that all directories get marked set-gid, that no files are other-writable, that both are user-writable and group-writable, and that both have consistent executability across all bits:

```
--chmod=Dg+s,ug+w,Fo-w,+X
```

Using octal mode numbers is also allowed:

```
--chmod=D2775,F664
```

It is also legal to specify multiple **--chmod** options, as each additional option is just appended to the list of changes to make.

See the **--perms** and **--executability** options for how the resulting permission value can be applied to the files in the transfer.

--owner, -o

This option causes rsync to set the owner of the destination file to be the same as the source file, but only if the receiving rsync is being run as the super-user (see also the **--super** and **--fake-super** options). Without this option, the owner of new and/or transferred files are set to the invoking user on the receiving side.

The preservation of ownership will associate matching names by default, but may fall back to using the ID number in some circumstances (see also the **--numeric-ids** option for a full discussion).

--group, -g

This option causes rsync to set the group of the destination file to be the same as the source file. If the receiving program is not running as the super-user (or if **--no-super** was specified), only groups that the invoking user on the receiving side is a member of will be preserved. Without this option, the group is set to the default group of the invoking user on the receiving side.

The preservation of group information will associate matching names by default, but may fall back to using the ID number in some circumstances (see also the **--numeric-ids** option for a full discussion).

--devices

This option causes rsync to transfer character and block device files to the remote system to recreate these devices. This option has no effect if the receiving rsync is not run as the super-user (see also the **--super** and **--fake-super** options).

--specials

This option causes rsync to transfer special files such as named sockets and fifos.

-D The **-D** option is equivalent to **--devices --specials**.

--write-devices

This tells rsync to treat a device on the receiving side as a regular file, allowing the writing of file data into a device.

This option implies the **--inplace** option.

Be careful using this, as you should know what devices are present on the receiving side of the transfer, especially if running rsync as root.

This option is refused by an rsync daemon.

--times, -t

This tells rsync to transfer modification times along with the files and update them on the remote system. Note that if this option is not used, the optimization that excludes files that have not been modified cannot be effective; in other words, a missing **-t** or **-a** will cause the next transfer to behave as if it used **-I**, causing all files to be updated (though rsync's delta-transfer algorithm will make the update fairly efficient if the files haven't actually changed, you're much better off using **-t**).

--atimes, -U

This tells rsync to set the access (use) times of the destination files to the same value as the source files.

If repeated, it also sets the **--open-noatime** option, which can help you to make the sending and receiving systems have the same access times on the transferred files without needing to run rsync an extra time after a file is transferred.

Note that some older rsync versions (prior to 3.2.0) may have been built with a pre-release **--atimes** patch that does not imply **--open-noatime** when this option is repeated.

--open-noatime

This tells rsync to open files with the O_NOATIME flag (on systems that support it) to avoid changing the access time of the files that are being transferred. If your OS does not support the O_NOATIME flag then rsync will silently ignore this option. Note also that some filesystems are mounted to avoid updating the atime on read access even without the O_NOATIME flag being set.

--crtimes, -N,

This tells rsync to set the create times (newness) of the destination files to the same value as the source files.

--omit-dir-times, -O

This tells rsync to omit directories when it is preserving modification times (see **--times**). If NFS is sharing the directories on the receiving side, it is a good idea to use **-O**. This option is inferred if you use **--backup** without **--backup-dir**.

This option also has the side-effect of avoiding early creation of directories in incremental recursion copies. The default **--inc-recursive** copying normally does an early-create pass of all the sub-directories in a parent directory in order for it to be able to then set the modify time of the parent directory right away (without having to delay that until a bunch of recursive copying has finished). This early-create idiom is not necessary if directory modify times are not being preserved, so it is skipped. Since early-create directories don't have accurate mode, mtime, or ownership, the use of this option can help when someone wants to avoid these partially-finished directories.

--omit-link-times, -J

This tells rsync to omit symlinks when it is preserving modification times (see **--times**).

--super

This tells the receiving side to attempt super-user activities even if the receiving rsync wasn't run by the super-user. These activities include: preserving users via the **--owner** option, preserving all groups (not just the current user's groups) via the **--group** option, and copying devices via the **--devices** option. This is useful for systems that allow such activities without being the super-user, and also for ensuring that you will get errors if the receiving side isn't being run as the super-user. To turn off super-user activities, the super-user can use **--no-super**.

--fake-super

When this option is enabled, rsync simulates super-user activities by saving/restoring the privileged attributes via special extended attributes that are attached to each file (as needed). This includes the file's owner and group (if it is not the default), the file's device info (device & special files are created as empty text files), and any permission bits that we won't allow to be set on the real file (e.g. the real file gets u-s,g-s,o-t for safety) or that would limit the owner's access (since

the real super-user can always access/change a file, the files we create can always be accessed/changed by the creating user). This option also handles ACLs (if **--acls** was specified) and non-user extended attributes (if **--xattrs** was specified).

This is a good way to backup data without using a super-user, and to store ACLs from incompatible systems.

The **--fake-super** option only affects the side where the option is used. To affect the remote side of a remote-shell connection, use the **--remote-option (-M)** option:

```
rsync -av -M--fake-super /src/ host:/dest/
```

For a local copy, this option affects both the source and the destination. If you wish a local copy to enable this option just for the destination files, specify **-M--fake-super**. If you wish a local copy to enable this option just for the source files, combine **--fake-super** with **-M--super**.

This option is overridden by both **--super** and **--no-super**.

See also the "fake super" setting in the daemon's rsyncd.conf file.

--sparse, -S

Try to handle sparse files efficiently so they take up less space on the destination. If combined with **--inplace** the file created might not end up with sparse blocks with some combinations of kernel version and/or filesystem type. If **--whole-file** is in effect (e.g. for a local copy) then it will always work because rsync truncates the file prior to writing out the updated version.

Note that versions of rsync older than 3.1.3 will reject the combination of **--sparse** and **--inplace**.

--preallocate

This tells the receiver to allocate each destination file to its eventual size before writing data to the file. Rsync will only use the real filesystem-level preallocation support provided by Linux's **fallocate(2)** system call or Cygwin's **posix_fallocate(3)**, not the slow glibc implementation that writes a null byte into each block.

Without this option, larger files may not be entirely contiguous on the filesystem, but with this option rsync will probably copy more slowly. If the destination is not an extent-supporting filesystem (such as ext4, xfs, NTFS, etc.), this option may have no positive effect at all.

If combined with **--sparse**, the file will only have sparse blocks (as opposed to allocated sequences of null bytes) if the kernel version and filesystem type support creating holes in the allocated data.

--dry-run, -n

This makes rsync perform a trial run that doesn't make any changes (and produces mostly the same output as a real run). It is most commonly used in combination with the **--verbose, -v** and/or **--itemize-changes, -i** options to see what an rsync command is going to do before one actually runs it.

The output of **--itemize-changes** is supposed to be exactly the same on a dry run and a subsequent real run (barring intentional trickery and system call failures); if it isn't, that's a bug. Other output should be mostly unchanged, but may differ in some areas. Notably, a dry run does not send the actual data for file transfers, so **--progress** has no effect, the "bytes sent", "bytes received", "literal data", and "matched data" statistics are too small, and the "speedup" value is equivalent to a run where no file transfers were needed.

--whole-file, -W

This option disables rsync's delta-transfer algorithm, which causes all transferred files to be sent whole. The transfer may be faster if this option is used when the bandwidth between the source and destination machines is higher than the bandwidth to disk (especially when the "disk" is actually a networked filesystem). This is the default when both the source and destination are specified as local paths, but only if no batch-writing option is in effect.

--checksum-choice=STR, --cc=STR

This option overrides the checksum algorithms. If one algorithm name is specified, it is used for both the transfer checksums and (assuming **--checksum** is specified) the pre-transfer checksums. If two comma-separated names are supplied, the first name affects the transfer checksums, and the second name affects the pre-transfer checksums (**-c**).

The checksum options that you may be able to use are:

- o **auto** (the default automatic choice)
- o **xxh128**
- o **xxh3**
- o **xxh64** (aka **xxhash**)
- o **md5**
- o **md4**
- o **none**

Run **rsync --version** to see the default checksum list compiled into your version (which may differ from the list above).

If "none" is specified for the first (or only) name, the **--whole-file** option is forced on and no checksum verification is performed on the transferred data. If "none" is specified for the second (or only) name, the **--checksum** option cannot be used.

The "auto" option is the default, where rsync bases its algorithm choice on a negotiation between the client and the server as follows:

When both sides of the transfer are at least 3.2.0, rsync chooses the first algorithm in the client's list of choices that is also in the server's list of choices. If no common checksum choice is found, rsync exits with an error. If the remote rsync is too old to support checksum negotiation, a value is chosen based on the protocol version (which chooses between MD5 and various flavors of MD4 based on protocol age).

The default order can be customized by setting the environment variable **RSYNC_CHECKSUM_LIST** to a space-separated list of acceptable checksum names. If the string contains a "&" character, it is separated into the "client string & server string", otherwise the same string applies to both. If the string (or string portion) contains no non-whitespace characters, the default checksum list is used. This method does not allow you to specify the transfer checksum separately from the pre-transfer checksum, and it discards "auto" and all unknown checksum names. A list with only invalid names results in a failed negotiation.

The use of the **--checksum-choice** option overrides this environment list.

--one-file-system, -x

This tells rsync to avoid crossing a filesystem boundary when recursing. This does not limit the user's ability to specify items to copy from multiple filesystems, just rsync's recursion through the hierarchy of each directory that the user specified, and also the analogous recursion on the receiving side during deletion. Also keep in mind that rsync treats a "bind" mount to the same device as being on the same filesystem.

If this option is repeated, rsync omits all mount-point directories from the copy. Otherwise, it includes an empty directory at each mount-point it encounters (using the attributes of the mounted directory because those of the underlying mount-point directory are inaccessible).

If rsync has been told to collapse symlinks (via **--copy-links** or **--copy-unsafe-links**), a symlink to a directory on another device is treated like a mount-point. Symlinks to non-directories are unaffected by this option.

--existing, --ignore-non-existing

This tells rsync to skip creating files (including directories) that do not exist yet on the destination. If this option is combined with the **--ignore-existing** option, no files will be updated (which can be useful if all you want to do is delete extraneous files).

This option is a transfer rule, not an exclude, so it doesn't affect the data that goes into the file-lists, and thus it doesn't affect deletions. It just limits the files that the receiver requests to be transferred.

--ignore-existing

This tells rsync to skip updating files that already exist on the destination (this does *not* ignore existing directories, or nothing would get done). See also **--existing**.

This option is a transfer rule, not an exclude, so it doesn't affect the data that goes into the file-lists, and thus it doesn't affect deletions. It just limits the files that the receiver requests to be transferred.

This option can be useful for those doing backups using the **--link-dest** option when they need to continue a backup run that got interrupted. Since a **--link-dest** run is copied into a new directory hierarchy (when it is used properly), using **--ignore-existing** will ensure that the already-handled files don't get tweaked (which avoids a change in permissions on the hard-linked files). This does mean that this option is only looking at the existing files in the destination hierarchy itself.

--remove-source-files

This tells rsync to remove from the sending side the files (meaning non-directories) that are a part of the transfer and have been successfully duplicated on the receiving side.

Note that you should only use this option on source files that are quiescent. If you are using this to move files that show up in a particular directory over to another host, make sure that the finished files get renamed into the source directory, not directly written into it, so that rsync can't possibly transfer a file that is not yet fully written. If you can't first write the files into a different directory, you should use a naming idiom that lets rsync avoid transferring files that are not yet finished (e.g. name the file "foo.new" when it is written, rename it to "foo" when it is done, and then use the option **--exclude='*.new'** for the rsync transfer).

Starting with 3.1.0, rsync will skip the sender-side removal (and output an error) if the file's size or modify time has not stayed unchanged.

--delete

This tells rsync to delete extraneous files from the receiving side (ones that aren't on the sending side), but only for the directories that are being synchronized. You must have asked rsync to send the whole directory (e.g. "**dir**" or "**dir/**") without using a wildcard for the directory's contents (e.g. "**dir/***") since the wildcard is expanded by the shell and rsync thus gets a request to transfer individual files, not the files' parent directory. Files that are excluded from the transfer are also excluded from being deleted unless you use the **--delete-excluded** option or mark the rules as only matching on the sending side (see the include/exclude modifiers in the FILTER RULES section).

Prior to rsync 2.6.7, this option would have no effect unless **--recursive** was enabled. Beginning with 2.6.7, deletions will also occur when **--dirs (-d)** is enabled, but only for directories whose contents are being copied.

This option can be dangerous if used incorrectly! It is a very good idea to first try a run using the **--dry-run** option (**-n**) to see what files are going to be deleted.

If the sending side detects any I/O errors, then the deletion of any files at the destination will be automatically disabled. This is to prevent temporary filesystem failures (such as NFS errors) on the sending side from causing a massive deletion of files on the destination. You can override this with the **--ignore-errors** option.

The **--delete** option may be combined with one of the **--delete-WHEN** options without conflict, as well as **--delete-excluded**. However, if none of the **--delete-WHEN** options are specified, rsync will choose the **--delete-during** algorithm when talking to rsync 3.0.0 or newer, and the

--delete-before algorithm when talking to an older rsync. See also **--delete-delay** and **--delete-after**.

--delete-before

Request that the file-deletions on the receiving side be done before the transfer starts. See **--delete** (which is implied) for more details on file-deletion.

Deleting before the transfer is helpful if the filesystem is tight for space and removing extraneous files would help to make the transfer possible. However, it does introduce a delay before the start of the transfer, and this delay might cause the transfer to timeout (if **--timeout** was specified). It also forces rsync to use the old, non-incremental recursion algorithm that requires rsync to scan all the files in the transfer into memory at once (see **--recursive**).

--delete-during, --del

Request that the file-deletions on the receiving side be done incrementally as the transfer happens. The per-directory delete scan is done right before each directory is checked for updates, so it behaves like a more efficient **--delete-before**, including doing the deletions prior to any per-directory filter files being updated. This option was first added in rsync version 2.6.4. See **--delete** (which is implied) for more details on file-deletion.

--delete-delay

Request that the file-deletions on the receiving side be computed during the transfer (like **--delete-during**), and then removed after the transfer completes. This is useful when combined with **--delay-updates** and/or **--fuzzy**, and is more efficient than using **--delete-after** (but can behave differently, since **--delete-after** computes the deletions in a separate pass after all updates are done). If the number of removed files overflows an internal buffer, a temporary file will be created on the receiving side to hold the names (it is removed while open, so you shouldn't see it during the transfer). If the creation of the temporary file fails, rsync will try to fall back to using **--delete-after** (which it cannot do if **--recursive** is doing an incremental scan). See **--delete** (which is implied) for more details on file-deletion.

--delete-after

Request that the file-deletions on the receiving side be done after the transfer has completed. This is useful if you are sending new per-directory merge files as a part of the transfer and you want their exclusions to take effect for the delete phase of the current transfer. It also forces rsync to use the old, non-incremental recursion algorithm that requires rsync to scan all the files in the transfer into memory at once (see **--recursive**). See **--delete** (which is implied) for more details on file-deletion.

--delete-excluded

In addition to deleting the files on the receiving side that are not on the sending side, this tells rsync to also delete any files on the receiving side that are excluded (see **--exclude**). See the FILTER RULES section for a way to make individual exclusions behave this way on the receiver, and for a way to protect files from **--delete-excluded**. See **--delete** (which is implied) for more details on file-deletion.

--ignore-missing-args

When rsync is first processing the explicitly requested source files (e.g. command-line arguments or **--files-from** entries), it is normally an error if the file cannot be found. This option suppresses that error, and does not try to transfer the file. This does not affect subsequent vanished-file errors if a file was initially found to be present and later is no longer there.

--delete-missing-args

This option takes the behavior of (the implied) **--ignore-missing-args** option a step farther: each missing arg will become a deletion request of the corresponding destination file on the receiving side (should it exist). If the destination file is a non-empty directory, it will only be successfully deleted if **--force** or **--delete** are in effect. Other than that, this option is independent of any other type of delete processing.

The missing source files are represented by special file-list entries which display as a "***missing**" entry in the **--list-only** output.

--ignore-errors

Tells **--delete** to go ahead and delete files even when there are I/O errors.

--force

This option tells rsync to delete a non-empty directory when it is to be replaced by a non-directory. This is only relevant if deletions are not active (see **--delete** for details).

Note for older rsync versions: **--force** used to still be required when using **--delete-after**, and it used to be non-functional unless the **--recursive** option was also enabled.

--max-delete=NUM

This tells rsync not to delete more than NUM files or directories. If that limit is exceeded, all further deletions are skipped through the end of the transfer. At the end, rsync outputs a warning (including a count of the skipped deletions) and exits with an error code of 25 (unless some more important error condition also occurred).

Beginning with version 3.0.0, you may specify **--max-delete=0** to be warned about any extraneous files in the destination without removing any of them. Older clients interpreted this as "unlimited", so if you don't know what version the client is, you can use the less obvious **--max-delete=-1** as a backward-compatible way to specify that no deletions be allowed (though really old versions didn't warn when the limit was exceeded).

--max-size=SIZE

This tells rsync to avoid transferring any file that is larger than the specified SIZE. A numeric value can be suffixed with a string to indicate the numeric units or left unqualified to specify bytes. Feel free to use a fractional value along with the units, such as **--max-size=1.5m**.

This option is a transfer rule, not an exclude, so it doesn't affect the data that goes into the file-lists, and thus it doesn't affect deletions. It just limits the files that the receiver requests to be transferred.

The first letter of a units string can be **B** (bytes), **K** (kilo), **M** (mega), **G** (giga), **T** (tera), or **P** (peta). If the string is a single char or has "ib" added to it (e.g. "G" or "GiB") then the units are multiples of 1024. If you use a two-letter suffix that ends with a "B" (e.g. "kb") then you get units that are multiples of 1000. The string's letters can be any mix of upper and lower-case that you want to use.

Finally, if the string ends with either "+1" or "-1", it is offset by one byte in the indicated direction. The largest possible value is usually **8192P-1**.

Examples: **--max-size=1.5mb-1** is 1499999 bytes, and **--max-size=2g+1** is 2147483649 bytes.

Note that rsync versions prior to 3.1.0 did not allow **--max-size=0**.

--min-size=SIZE

This tells rsync to avoid transferring any file that is smaller than the specified SIZE, which can help in not transferring small, junk files. See the **--max-size** option for a description of SIZE and other information.

Note that rsync versions prior to 3.1.0 did not allow **--min-size=0**.

--max-alloc=SIZE

By default rsync limits an individual malloc/realloc to about 1GB in size. For most people this limit works just fine and prevents a protocol error causing rsync to request massive amounts of memory. However, if you have many millions of files in a transfer, a large amount of server memory, and you don't want to split up your transfer into multiple parts, you can increase the per-allocation limit to something larger and rsync will consume more memory.

Keep in mind that this is not a limit on the total size of allocated memory. It is a sanity-check value for each individual allocation.

See the **--max-size** option for a description of how SIZE can be specified. The default suffix if none is given is bytes.

Beginning in 3.2.3, a value of 0 specifies no limit.

You can set a default value using the environment variable RSYNC_MAX_ALLOC using the same SIZE values as supported by this option. If the remote rsync doesn't understand the **--max-alloc** option, you can override an environmental value by specifying **--max-alloc=1g**, which will make rsync avoid sending the option to the remote side (because "1G" is the default).

--block-size=SIZE, -B

This forces the block size used in rsync's delta-transfer algorithm to a fixed value. It is normally selected based on the size of each file being updated. See the technical report for details.

Beginning in 3.2.3 the SIZE can be specified with a suffix as detailed in the **--max-size** option. Older versions only accepted a byte count.

--rsh=COMMAND, -e

This option allows you to choose an alternative remote shell program to use for communication between the local and remote copies of rsync. Typically, rsync is configured to use ssh by default, but you may prefer to use rsh on a local network.

If this option is used with **[user@]host::module/path**, then the remote shell *COMMAND* will be used to run an rsync daemon on the remote host, and all data will be transmitted through that remote shell connection, rather than through a direct socket connection to a running rsync daemon on the remote host. See the section "USING RSYNC-DAEMON FEATURES VIA A REMOTE-SHELL CONNECTION" above.

Beginning with rsync 3.2.0, the RSYNC_PORT environment variable will be set when a daemon connection is being made via a remote-shell connection. It is set to 0 if the default daemon port is being assumed, or it is set to the value of the rsync port that was specified via either the **--port** option or a non-empty port value in an rsync:// URL. This allows the script to discern if a non-default port is being requested, allowing for things such as an SSL or stunnel helper script to connect to a default or alternate port.

Command-line arguments are permitted in COMMAND provided that COMMAND is presented to rsync as a single argument. You must use spaces (not tabs or other whitespace) to separate the command and args from each other, and you can use single- and/or double-quotes to preserve spaces in an argument (but not backslashes). Note that doubling a single-quote inside a single-quoted string gives you a single-quote; likewise for double-quotes (though you need to pay attention to which quotes your shell is parsing and which quotes rsync is parsing). Some examples:

```
-e 'ssh -p 2234'  
-e 'ssh -o "ProxyCommand nohup ssh firewall nc -w1 %h %p"'
```

(Note that ssh users can alternately customize site-specific connect options in their .ssh/config file.)

You can also choose the remote shell program using the RSYNC_RSH environment variable, which accepts the same range of values as **-e**.

See also the **--blocking-io** option which is affected by this option.

--rsync-path=PROGRAM

Use this to specify what program is to be run on the remote machine to start-up rsync. Often used when rsync is not in the default remote-shell's path (e.g. **--rsync-path=/usr/local/bin/rsync**). Note that PROGRAM is run with the help of a shell, so it can be any program, script, or command sequence you'd care to run, so long as it does not corrupt the standard-in & standard-out that rsync is using to communicate.

One tricky example is to set a different default directory on the remote machine for use with the **--relative** option. For instance:

```
rsync -avR --rsync-path="cd /a/b && rsync" host:c/d /e/
```

--remote-option=OPTION, -M

This option is used for more advanced situations where you want certain effects to be limited to one side of the transfer only. For instance, if you want to pass **--log-file=FILE** and **--fake-super** to the remote system, specify it like this:

```
rsync -av -M --log-file=foo -M--fake-super src/ dest/
```

If you want to have an option affect only the local side of a transfer when it normally affects both sides, send its negation to the remote side. Like this:

```
rsync -av -x -M--no-x src/ dest/
```

Be cautious using this, as it is possible to toggle an option that will cause rsync to have a different idea about what data to expect next over the socket, and that will make it fail in a cryptic fashion.

Note that it is best to use a separate **--remote-option** for each option you want to pass. This makes your usage compatible with the **--protect-args** option. If that option is off, any spaces in your remote options will be split by the remote shell unless you take steps to protect them.

When performing a local transfer, the "local" side is the sender and the "remote" side is the receiver.

Note some versions of the popt option-parsing library have a bug in them that prevents you from using an adjacent arg with an equal in it next to a short option letter (e.g. **-M--log-file=/tmp/foo**). If this bug affects your version of popt, you can use the version of popt that is included with rsync.

--cvs-exclude, -C

This is a useful shorthand for excluding a broad range of files that you often don't want to transfer between systems. It uses a similar algorithm to CVS to determine if a file should be ignored.

The exclude list is initialized to exclude the following items (these initial items are marked as perishable -- see the FILTER RULES section):

```
RCS SCCS CVS CVS.adm RCSLOG cvslog.* tags TAGS .make.state .nse_depinf~ #*  
.##,* _$* *$ *.old *.bak *.BAK *.orig *.rej .del-* *.a *.olb *.o *.obj *.so *.exe *.Z *.elc  
*.ln core .svn/ .git/ .hg/ .bzr/
```

then, files listed in a \$HOME/.cvsignore are added to the list and any files listed in the CVSIGNORE environment variable (all cvsignore names are delimited by whitespace).

Finally, any file is ignored if it is in the same directory as a .cvsignore file and matches one of the patterns listed therein. Unlike rsync's filter/exclude files, these patterns are split on whitespace. See the **cvs(1)** manual for more information.

If you're combining **-C** with your own **--filter** rules, you should note that these CVS excludes are appended at the end of your own rules, regardless of where the **-C** was placed on the command-line. This makes them a lower priority than any rules you specified explicitly. If you want to control where these CVS excludes get inserted into your filter rules, you should omit the **-C** as a command-line option and use a combination of **--filter=:C** and **--filter=-C** (either on your command-line or by putting the ":C" and "-C" rules into a filter file with your other rules). The first option turns on the per-directory scanning for the .cvsignore file. The second option does a one-time import of the CVS excludes mentioned above.

--filter=RULE, -f

This option allows you to add rules to selectively exclude certain files from the list of files to be transferred. This is most useful in combination with a recursive transfer.

You may use as many **--filter** options on the command line as you like to build up the list of files to exclude. If the filter contains whitespace, be sure to quote it so that the shell gives the rule to rsync as a single argument. The text below also mentions that you can use an underscore to replace the space that separates a rule from its arg.

See the FILTER RULES section for detailed information on this option.

- F** The **-F** option is a shorthand for adding two **--filter** rules to your command. The first time it is used is a shorthand for this rule:

```
--filter='dir-merge .rsync-filter'
```

This tells rsync to look for per-directory .rsync-filter files that have been sprinkled through the hierarchy and use their rules to filter the files in the transfer. If **-F** is repeated, it is a shorthand for this rule:

```
--filter='exclude .rsync-filter'
```

This filters out the .rsync-filter files themselves from the transfer.

See the FILTER RULES section for detailed information on how these options work.

--exclude=PATTERN

This option is a simplified form of the **--filter** option that defaults to an exclude rule and does not allow the full rule-parsing syntax of normal filter rules.

See the FILTER RULES section for detailed information on this option.

--exclude-from=FILE

This option is related to the **--exclude** option, but it specifies a FILE that contains exclude patterns (one per line). Blank lines in the file are ignored, as are whole-line comments that start with ';' or '#' (filename rules that contain those characters are unaffected).

If *FILE* is '**-**', the list will be read from standard input.

--include=PATTERN

This option is a simplified form of the **--filter** option that defaults to an include rule and does not allow the full rule-parsing syntax of normal filter rules.

See the FILTER RULES section for detailed information on this option.

--include-from=FILE

This option is related to the **--include** option, but it specifies a FILE that contains include patterns (one per line). Blank lines in the file are ignored, as are whole-line comments that start with ';' or '#' (filename rules that contain those characters are unaffected).

If *FILE* is '**-**', the list will be read from standard input.

--files-from=FILE

Using this option allows you to specify the exact list of files to transfer (as read from the specified FILE or '**-**' for standard input). It also tweaks the default behavior of rsync to make transferring just the specified files and directories easier:

- o The **--relative (-R)** option is implied, which preserves the path information that is specified for each item in the file (use **--no-relative** or **--no-R** if you want to turn that off).
- o The **--dirs (-d)** option is implied, which will create directories specified in the list on the destination rather than noisily skipping them (use **--no-dirs** or **--no-d** if you want to turn that off).
- o The **--archive (-a)** option's behavior does not imply **--recursive (-r)**, so specify it explicitly, if you want it.
- o These side-effects change the default state of rsync, so the position of the **--files-from** option on the command-line has no bearing on how other options are parsed (e.g. **-a** works the same before or after **--files-from**, as does **--no-R** and all other options).

The filenames that are read from the FILE are all relative to the source dir — any leading slashes are removed and no ".." references are allowed to go higher than the source dir. For example, take this command:

```
rsync -a --files-from=/tmp/foo /usr remote:/backup
```

If /tmp/foo contains the string "bin" (or even "/bin"), the /usr/bin directory will be created as /backup/bin on the remote host. If it contains "bin/" (note the trailing slash), the immediate contents of the directory would also be sent (without needing to be explicitly mentioned in the file — this began in version 2.6.4). In both cases, if the **-r** option was enabled, that dir's entire hierarchy would also be transferred (keep in mind that **-r** needs to be specified explicitly with **--files-from**, since it is not implied by **-a**). Also note that the effect of the (enabled by default) **--relative** option is to duplicate only the path info that is read from the file — it does not force the duplication of the source-spec path (/usr in this case).

In addition, the **--files-from** file can be read from the remote host instead of the local host if you specify a "host:" in front of the file (the host must match one end of the transfer). As a short-cut, you can specify just a prefix of ":" to mean "use the remote end of the transfer". For example:

```
rsync -a --files-from=:/path/file-list src:/ /tmp/copy
```

This would copy all the files specified in the /path/file-list file that was located on the remote "src" host.

If the **--iconv** and **--protect-args** options are specified and the **--files-from** filenames are being sent from one host to another, the filenames will be translated from the sending host's charset to the receiving host's charset.

NOTE: sorting the list of files in the **--files-from** input helps rsync to be more efficient, as it will avoid re-visiting the path elements that are shared between adjacent entries. If the input is not sorted, some path elements (implied directories) may end up being scanned multiple times, and rsync will eventually unduplicate them after they get turned into file-list elements.

--from0, -0

This tells rsync that the rules/filenames it reads from a file are terminated by a null ('\0') character, not a NL, CR, or CR+LF. This affects **--exclude-from**, **--include-from**, **--files-from**, and any merged files specified in a **--filter** rule. It does not affect **--cvs-exclude** (since all names read from a .cvignore file are split on whitespace).

--protect-args, -s

This option sends all filenames and most options to the remote rsync without allowing the remote shell to interpret them. This means that spaces are not split in names, and any non-wildcard special characters are not translated (such as ~, \$, ;, &, etc.). Wildcards are expanded on the remote host by rsync (instead of the shell doing it).

If you use this option with **--iconv**, the args related to the remote side will also be translated from the local to the remote character-set. The translation happens before wild-cards are expanded. See also the **--files-from** option.

You may also control this option via the RSYNC_PROTECT_ARGS environment variable. If this variable has a non-zero value, this option will be enabled by default, otherwise it will be disabled by default. Either state is overridden by a manually specified positive or negative version of this option (note that **--no-s** and **--no-protect-args** are the negative versions). Since this option was first introduced in 3.0.0, you'll need to make sure it's disabled if you ever need to interact with a remote rsync that is older than that.

Rsync can also be configured (at build time) to have this option enabled by default (with is overridden by both the environment and the command-line). Run **rsync --version** to check if this is the case, as it will display "default protect-args" or "optional protect-args" depending on how it was compiled.

This option will eventually become a new default setting at some as-yet-undetermined point in the future.

--copy-as=USER[:GROUP]

This option instructs rsync to use the USER and (if specified after a colon) the GROUP for the copy operations. This only works if the user that is running rsync has the ability to change users. If the group is not specified then the user's default groups are used.

This option can help to reduce the risk of an rsync being run as root into or out of a directory that might have live changes happening to it and you want to make sure that root-level read or write actions of system files are not possible. While you could alternatively run all of rsync as the specified user, sometimes you need the root-level host-access credentials to be used, so this allows rsync to drop root for the copying part of the operation after the remote-shell or daemon connection is established.

The option only affects one side of the transfer unless the transfer is local, in which case it affects both sides. Use the **--remote-option** to affect the remote side, such as **-M--copy-as=joe**. For a local transfer, the lsh (or lsh.sh) support file provides a local-shell helper script that can be used to allow a "localhost:" or "lh:" host-spec to be specified without needing to setup any remote shells, allowing you to specify remote options that affect the side of the transfer that is using the host-spec (and using hostname "lh" avoids the overriding of the remote directory to the user's home dir).

For example, the following rsync writes the local files as user "joe":

```
sudo rsync -aiv --copy-as=joe host1:backups/joe/ /home/joe/
```

This makes all files owned by user "joe", limits the groups to those that are available to that user, and makes it impossible for the joe user to do a timed exploit of the path to induce a change to a file that the joe user has no permissions to change.

The following command does a local copy into the "dest/" dir as user "joe" (assuming you've installed support/lsh into a dir on your \$PATH):

```
sudo rsync -aive lsh -M--copy-as=joe src/ lh:dest/
```

--temp-dir=DIR, -T

This option instructs rsync to use DIR as a scratch directory when creating temporary copies of the files transferred on the receiving side. The default behavior is to create each temporary file in the same directory as the associated destination file. Beginning with rsync 3.1.1, the temp-file names inside the specified DIR will not be prefixed with an extra dot (though they will still have a random suffix added).

This option is most often used when the receiving disk partition does not have enough free space to hold a copy of the largest file in the transfer. In this case (i.e. when the scratch directory is on a different disk partition), rsync will not be able to rename each received temporary file over the top of the associated destination file, but instead must copy it into place. Rsync does this by copying the file over the top of the destination file, which means that the destination file will contain truncated data during this copy. If this were not done this way (even if the destination file were first removed, the data locally copied to a temporary file in the destination directory, and then renamed into place) it would be possible for the old file to continue taking up disk space (if someone had it open), and thus there might not be enough room to fit the new version on the disk at the same time.

If you are using this option for reasons other than a shortage of disk space, you may wish to combine it with the **--delay-updates** option, which will ensure that all copied files get put into subdirectories in the destination hierarchy, awaiting the end of the transfer. If you don't have enough room to duplicate all the arriving files on the destination partition, another way to tell rsync that you aren't overly concerned about disk space is to use the **--partial-dir** option with a relative path; because this tells rsync that it is OK to stash off a copy of a single file in a subdir in the destination hierarchy, rsync will use the partial-dir as a staging area to bring over the copied file, and then rename it into place from there. (Specifying a **--partial-dir** with an absolute path does not have this side-effect.)

--fuzzy, -y

This option tells rsync that it should look for a basis file for any destination file that is missing. The current algorithm looks in the same directory as the destination file for either a file that has an identical size and modified-time, or a similarly-named file. If found, rsync uses the fuzzy basis file to try to speed up the transfer.

If the option is repeated, the fuzzy scan will also be done in any matching alternate destination directories that are specified via **--compare-dest**, **--copy-dest**, or **--link-dest**.

Note that the use of the **--delete** option might get rid of any potential fuzzy-match files, so either use **--delete-after** or specify some filename exclusions if you need to prevent this.

--compare-dest=DIR

This option instructs rsync to use *DIR* on the destination machine as an additional hierarchy to compare destination files against doing transfers (if the files are missing in the destination directory). If a file is found in *DIR* that is identical to the sender's file, the file will NOT be transferred to the destination directory. This is useful for creating a sparse backup of just files that have changed from an earlier backup. This option is typically used to copy into an empty (or newly created) directory.

Beginning in version 2.6.4, multiple **--compare-dest** directories may be provided, which will cause rsync to search the list in the order specified for an exact match. If a match is found that differs only in attributes, a local copy is made and the attributes updated. If a match is not found, a basis file from one of the *DIRs* will be selected to try to speed up the transfer.

If *DIR* is a relative path, it is relative to the destination directory. See also **--copy-dest** and **--link-dest**.

NOTE: beginning with version 3.1.0, rsync will remove a file from a non-empty destination hierarchy if an exact match is found in one of the compare-dest hierarchies (making the end result more closely match a fresh copy).

--copy-dest=DIR

This option behaves like **--compare-dest**, but rsync will also copy unchanged files found in *DIR* to the destination directory using a local copy. This is useful for doing transfers to a new destination while leaving existing files intact, and then doing a flash-cutover when all files have been successfully transferred.

Multiple **--copy-dest** directories may be provided, which will cause rsync to search the list in the order specified for an unchanged file. If a match is not found, a basis file from one of the *DIRs* will be selected to try to speed up the transfer.

If *DIR* is a relative path, it is relative to the destination directory. See also **--compare-dest** and **--link-dest**.

--link-dest=DIR

This option behaves like **--copy-dest**, but unchanged files are hard linked from *DIR* to the destination directory. The files must be identical in all preserved attributes (e.g. permissions, possibly ownership) in order for the files to be linked together. An example:

```
rsync -av --link-dest=$PWD/prior_dir host:src_dir/ new_dir/
```

If file's aren't linking, double-check their attributes. Also check if some attributes are getting forced outside of rsync's control, such a mount option that squishes root to a single user, or mounts a removable drive with generic ownership (such as OS X's "Ignore ownership on this volume" option).

Beginning in version 2.6.4, multiple **--link-dest** directories may be provided, which will cause rsync to search the list in the order specified for an exact match (there is a limit of 20 such directories). If a match is found that differs only in attributes, a local copy is made and the attributes updated. If a match is not found, a basis file from one of the *DIRs* will be selected to try to speed up the transfer.

This option works best when copying into an empty destination hierarchy, as existing files may get their attributes tweaked, and that can affect alternate destination files via hard-links. Also, itemizing of changes can get a bit muddled. Note that prior to version 3.1.0, an alternate-directory exact match would never be found (nor linked into the destination) when a destination file already exists.

Note that if you combine this option with **--ignore-times**, rsync will not link any files together because it only links identical files together as a substitute for transferring the file, never as an additional check after the file is updated.

If *DIR* is a relative path, it is relative to the destination directory. See also **--compare** **e-dest** and **--copy-dest**.

Note that rsync versions prior to 2.6.1 had a bug that could prevent **--link-dest** from working properly for a non-super-user when **-o** was specified (or implied by **-a**). You can work-around this bug by avoiding the **-o** option when sending to an old rsync.

--compress, -z

With this option, rsync compresses the file data as it is sent to the destination machine, which reduces the amount of data being transmitted — something that is useful over a slow connection.

Rsync supports multiple compression methods and will choose one for you unless you force the choice using the **--compress-choice** (**--zc**) option.

Run **rsync --version** to see the default compress list compiled into your version.

When both sides of the transfer are at least 3.2.0, rsync chooses the first algorithm in the client's list of choices that is also in the server's list of choices. If no common compress choice is found, rsync exits with an error. If the remote rsync is too old to support checksum negotiation, its list is assumed to be "zlib".

The default order can be customized by setting the environment variable **RSYNC_COMPRESS_LIST** to a space-separated list of acceptable compression names. If the string contains a "&" character, it is separated into the "client string & server string", otherwise the same string applies to both. If the string (or string portion) contains no non-whitespace characters, the default compress list is used. Any unknown compression names are discarded from the list, but a list with only invalid names results in a failed negotiation.

There are some older rsync versions that were configured to reject a **-z** option and require the use of **-zz** because their compression library was not compatible with the default zlib compression method. You can usually ignore this weirdness unless the rsync server complains and tells you to specify **-zz**.

See also the **--skip-compress** option for the default list of file suffixes that will be transferred with no (or minimal) compression.

--compress-choice=STR, --zc=STR

This option can be used to override the automatic negotiation of the compression algorithm that occurs when **--compress** is used. The option implies **--compress** unless "none" was specified, which instead implies **--no-compress**.

The compression options that you may be able to use are:

- o **zstd**
- o **lz4**
- o **zlibx**
- o **zlib**
- o **none**

Run **rsync --version** to see the default compress list compiled into your version (which may differ from the list above).

Note that if you see an error about an option named **--old-compress** or **--new-compress**, this is rsync trying to send the **--compress-choice=zlib** or **--compress-choice=zlibx** option in a backward-compatible manner that more rsync versions understand. This error indicates that the older rsync version on the server will not allow you to force the compression type.

Note that the "zlibx" compression algorithm is just the "zlib" algorithm with matched data excluded from the compression stream (to try to make it more compatible with an external zlib implementation).

--compress-level=NUM, --zl=NUM

Explicitly set the compression level to use (see **--compress**, **-z**) instead of letting it default. The **--compress** option is implied as long as the level chosen is not a "don't compress" level for the compression algorithm that is in effect (e.g. zlib compression treats level 0 as "off").

The level values vary depending on the checksum in effect. Because rsync will negotiate a checksum choice by default (when the remote rsync is new enough), it can be good to combine this option with a **--compress-choice** (**--zc**) option unless you're sure of the choice in effect. For example:

```
rsync -av --zc=zstd --zl=22 host:src/ dest/
```

For zlib & zlibx compression the valid values are from 1 to 9 with 6 being the default. Specifying 0 turns compression off, and specifying -1 chooses the default of 6.

For zstd compression the valid values are from -131072 to 22 with 3 being the default. Specifying 0 chooses the default of 3.

For lz4 compression there are no levels, so the value is always 0.

If you specify a too-large or too-small value, the number is silently limited to a valid value. This allows you to specify something like **--zl=999999999** and be assured that you'll end up with the maximum compression level no matter what algorithm was chosen.

If you want to know the compression level that is in effect, specify **--debug=nstr** to see the "negotiated string" results. This will report something like "**Client compress: zstd (level 3)**" (along with the checksum choice in effect).

--skip-compress=LIST

Override the list of file suffixes that will be compressed as little as possible. Rsync sets the compression level on a per-file basis based on the file's suffix. If the compression algorithm has an "off" level (such as zlib/zlibx) then no compression occurs for those files. Other algorithms that support changing the streaming level on-the-fly will have the level minimized to reduce the CPU usage as much as possible for a matching file. At this time, only zlib & zlibx compression support this changing of levels on a per-file basis.

The **LIST** should be one or more file suffixes (without the dot) separated by slashes (/). You may specify an empty string to indicate that no files should be skipped.

Simple character-class matching is supported: each must consist of a list of letters inside the square brackets (e.g. no special classes, such as "[[:alpha:]]", are supported, and '-' has no special meaning).

The characters asterisk (*) and question-mark (?) have no special meaning.

Here's an example that specifies 6 suffixes to skip (since 1 of the 5 rules matches 2 suffixes):

```
--skip-compress=gz/jpg/mp[34]/7z/bz2
```

The default file suffixes in the skip-compress list in this version of rsync are:

```
3g2 3gp 7z aac ace apk avi bz2 deb dmg ear f4v flac flv gpg gz iso jar jpeg jpg lrz lz lz4 lzma
lzo m1a m1v m2a m2ts m2v m4a m4b m4p m4r m4v mka mkv mov mp1 mp2 mp3 mp4 mpa
mpeg mpg mpv mts odb odf odi odm odp ods odt oga ogg ogm ogv ogx opus otg oth otp
ots ott oxt png qt rar rpm rz rzip spx squashfs sxc sxd sxa sgm sz tbz tbz2 tgz tlz ts txz
tzo vob war webm webp xz z zip zst
```

This list will be replaced by your **--skip-compress** list in all but one situation: a copy from a daemon rsync will add your skipped suffixes to its list of non-compressing files (and its list may be configured to a different default).

--numeric-ids

With this option rsync will transfer numeric group and user IDs rather than using user and group names and mapping them at both ends.

By default rsync will use the username and groupname to determine what ownership to give files. The special uid 0 and the special group 0 are never mapped via user/group names even if the **--numeric-ids** option is not specified.

If a user or group has no name on the source system or it has no match on the destination system, then the numeric ID from the source system is used instead. See also the comments on the "**use chroot**" setting in the rsyncd.conf manpage for information on how the chroot setting affects rsync's ability to look up the names of the users and groups and what you can do about it.

--usermap=STRING, --groupmap=STRING

These options allow you to specify users and groups that should be mapped to other values by the receiving side. The **STRING** is one or more **FROM:TO** pairs of values separated by commas. Any matching **FROM** value from the sender is replaced with a **TO** value from the receiver. You may specify usernames or user IDs for the **FROM** and **TO** values, and the **FROM** value may also be a wild-card string, which will be matched against the sender's names (wild-cards do NOT match against ID numbers, though see below for why a '*' matches everything). You may instead specify a range of ID numbers via an inclusive range: **LOW-HIGH**. For example:

```
--usermap=0-99:nobody,wayne:admin,*:normal --groupmap=usr:1,1:usr
```

The first match in the list is the one that is used. You should specify all your user mappings using a single **--usermap** option, and/or all your group mappings using a single **--groupmap** option.

Note that the sender's name for the 0 user and group are not transmitted to the receiver, so you should either match these values using a 0, or use the names in effect on the receiving side (typically "root"). All other **FROM** names match those in use on the sending side. All **TO** names match those in use on the receiving side.

Any IDs that do not have a name on the sending side are treated as having an empty name for the purpose of matching. This allows them to be matched via a "*" or using an empty name. For instance:

```
--usermap=:nobody --groupmap=:nobody
```

When the **--numeric-ids** option is used, the sender does not send any names, so all the IDs are treated as having an empty name. This means that you will need to specify numeric **FROM** values if you want to map these nameless IDs to different values.

For the **--usermap** option to have any effect, the **-o** (**--owner**) option must be used (or implied), and the receiver will need to be running as a super-user (see also the **--fake-super** option). For the **--groupmap** option to have any effect, the **-g** (**--group**) option must be used (or implied), and the receiver will need to have permissions to set that group.

If your shell complains about the wildcards, use **--protect-args** (**-s**).

--chown=USER:GROUP

This option forces all files to be owned by USER with group GROUP. This is a simpler interface than using **--usermap** and **--groupmap** directly, but it is implemented using those options internally, so you cannot mix them. If either the USER or GROUP is empty, no mapping for the omitted user/group will occur. If GROUP is empty, the trailing colon may be omitted, but if USER is empty, a leading colon must be supplied.

If you specify "**--chown=foo:bar**", this is exactly the same as specifying "**--usermap=*:foo --groupmap=*:bar**", only easier. If your shell complains about the wildcards, use **--protect-args** (**-s**).

--timeout=SECONDS

This option allows you to set a maximum I/O timeout in seconds. If no data is transferred for the specified time then rsync will exit. The default is 0, which means no timeout.

--contimeout=SECONDS

This option allows you to set the amount of time that rsync will wait for its connection to an rsync daemon to succeed. If the timeout is reached, rsync exits with an error.

--address=ADDRESS

By default rsync will bind to the wildcard address when connecting to an rsync daemon. The **--address** option allows you to specify a specific IP address (or hostname) to bind to. See also this option in the **--daemon** mode section.

--port=PORT

This specifies an alternate TCP port number to use rather than the default of 873. This is only needed if you are using the double-colon (::) syntax to connect with an rsync daemon (since the URL syntax has a way to specify the port as a part of the URL). See also this option in the **--daemon** mode section.

--sockopts=OPTIONS

This option can provide endless fun for people who like to tune their systems to the utmost degree. You can set all sorts of socket options which may make transfers faster (or slower!). Read the man page for the **setsockopt()** system call for details on some of the options you may be able to set. By default no special socket options are set. This only affects direct socket connections to a remote rsync daemon.

This option also exists in the **--daemon** mode section.

--blocking-io

This tells rsync to use blocking I/O when launching a remote shell transport. If the remote shell is either rsh or remsh, rsync defaults to using blocking I/O, otherwise it defaults to using non-blocking I/O. (Note that ssh prefers non-blocking I/O.)

--outbuf=MODE

This sets the output buffering mode. The mode can be None (aka Unbuffered), Line, or Block (aka Full). You may specify as little as a single letter for the mode, and use upper or lower case.

The main use of this option is to change Full buffering to Line buffering when rsync's output is going to a file or pipe.

--itemize-changes, -i

Requests a simple itemized list of the changes that are being made to each file, including attribute changes. This is exactly the same as specifying **--out-format='%i %n%L'**. If you repeat the option, unchanged files will also be output, but only if the receiving rsync is at least version 2.6.7 (you can use **-vv** with older versions of rsync, but that also turns on the output of other verbose messages).

The "%i" escape has a cryptic output that is 11 letters long. The general format is like the string **YXcstpqguax**, where **Y** is replaced by the type of update being done, **X** is replaced by the file-type, and the other letters represent attributes that may be output if they are being modified.

The update types that replace the **Y** are as follows:

- o A < means that a file is being transferred to the remote host (sent).
- o A > means that a file is being transferred to the local host (received).
- o A c means that a local change/creation is occurring for the item (such as the creation of a directory or the changing of a symlink, etc.).
- o A h means that the item is a hard link to another item (requires **--hard-links**).
- o A . means that the item is not being updated (though it might have attributes that are being modified).

- o A * means that the rest of the itemized-output area contains a message (e.g. "deleting").

The file-types that replace the **X** are: **f** for a file, a **d** for a directory, an **L** for a symlink, a **D** for a device, and a **S** for a special file (e.g. named sockets and fifos).

The other letters in the string indicate if some attributes of the file have changed, as follows:

- o "." – the attribute is unchanged.
- o "+" – the file is newly created.
- o " " – all the attributes are unchanged (all dots turn to spaces).
- o "?" – the change is unknown (when the remote rsync is old).
- o A letter indicates an attribute is being updated.

The attribute that is associated with each letter is as follows:

- o A **c** means either that a regular file has a different checksum (requires **--checksum**) or that a symlink, device, or special file has a changed value. Note that if you are sending files to an rsync prior to 3.0.1, this change flag will be present only for checksum-differing regular files.
- o A **s** means the size of a regular file is different and will be updated by the file transfer.
- o A **t** means the modification time is different and is being updated to the sender's value (requires **--times**). An alternate value of **T** means that the modification time will be set to the transfer time, which happens when a file/symlink/device is updated without **--times** and when a symlink is changed and the receiver can't set its time. (Note: when using an rsync 3.0.0 client, you might see the **s** flag combined with **t** instead of the proper **T** flag for this time-setting failure.)
- o A **p** means the permissions are different and are being updated to the sender's value (requires **--perms**).
- o An **o** means the owner is different and is being updated to the sender's value (requires **--owner** and super-user privileges).
- o A **g** means the group is different and is being updated to the sender's value (requires **--group** and the authority to set the group).
- o A **u|n|b** indicates the following information: **u** means the access (use) time is different and is being updated to the sender's value (requires **--atimes**); **n** means the create time (newness) is different and is being updated to the sender's value (requires **--crtimes**); **b** means that both the access and create times are being updated.
- o The **a** means that the ACL information is being changed.
- o The **x** means that the extended attribute information is being changed.

One other output is possible: when deleting files, the "%i" will output the string "***deleting**" for each item that is being removed (assuming that you are talking to a recent enough rsync that it logs deletions instead of outputting them as a verbose message).

--out-format=FORMAT

This allows you to specify exactly what the rsync client outputs to the user on a per-update basis. The format is a text string containing embedded single-character escape sequences prefixed with a percent (%) character. A default format of "%n%L" is assumed if either **--info=name** or **-v** is specified (this tells you just the name of the file and, if the item is a link, where it points). For a full list of the possible escape characters, see the "**log format**" setting in the rsyncd.conf manpage.

Specifying the **--out-format** option implies the **--info=name** option, which will mention each file, dir, etc. that gets updated in a significant way (a transferred file, a recreated symlink/device, or a touched directory). In addition, if the itemize-changes escape (%i) is included in the string (e.g. if the **--itemize-changes** option was used), the logging of names increases to mention any item

that is changed in any way (as long as the receiving side is at least 2.6.4). See the **--itemize-changes** option for a description of the output of "%i".

Rsync will output the out-format string prior to a file's transfer unless one of the transfer-statistic escapes is requested, in which case the logging is done at the end of the file's transfer. When this late logging is in effect and **--progress** is also specified, rsync will also output the name of the file being transferred prior to its progress information (followed, of course, by the out-format output).

--log-file=FILE

This option causes rsync to log what it is doing to a file. This is similar to the logging that a daemon does, but can be requested for the client side and/or the server side of a non-daemon transfer. If specified as a client option, transfer logging will be enabled with a default format of "%i %n%L". See the **--log-file-format** option if you wish to override this.

Here's a example command that requests the remote side to log what is happening:

```
rsync -av --remote-option=--log-file=/tmp/rlog src/ dest/
```

This is very useful if you need to debug why a connection is closing unexpectedly.

--log-file-format=FORMAT

This allows you to specify exactly what per-update logging is put into the file specified by the **--log-file** option (which must also be specified for this option to have any effect). If you specify an empty string, updated files will not be mentioned in the log file. For a list of the possible escape characters, see the "log format" setting in the rsyncd.conf manpage.

The default FORMAT used if **--log-file** is specified and this option is not is '%i %n%L'.

--stats This tells rsync to print a verbose set of statistics on the file transfer, allowing you to tell how effective rsync's delta-transfer algorithm is for your data. This option is equivalent to **--info=stats2** if combined with 0 or 1 **-v** options, or **--info=stats3** if combined with 2 or more **-v** options.

The current statistics are as follows:

- o **Number of files** is the count of all "files" (in the generic sense), which includes directories, symlinks, etc. The total count will be followed by a list of counts by filetype (if the total is non-zero). For example: "(reg: 5, dir: 3, link: 2, dev: 1, special: 1)" lists the totals for regular files, directories, symlinks, devices, and special files. If any of value is 0, it is completely omitted from the list.
- o **Number of created files** is the count of how many "files" (generic sense) were created (as opposed to updated). The total count will be followed by a list of counts by filetype (if the total is non-zero).
- o **Number of deleted files** is the count of how many "files" (generic sense) were deleted. The total count will be followed by a list of counts by filetype (if the total is non-zero). Note that this line is only output if deletions are in effect, and only if protocol 31 is being used (the default for rsync 3.1.x).
- o **Number of regular files transferred** is the count of normal files that were updated via rsync's delta-transfer algorithm, which does not include dirs, symlinks, etc. Note that rsync 3.1.0 added the word "regular" into this heading.
- o **Total file size** is the total sum of all file sizes in the transfer. This does not count any size for directories or special files, but does include the size of symlinks.
- o **Total transferred file size** is the total sum of all files sizes for just the transferred files.
- o **Literal data** is how much unmatched file-update data we had to send to the receiver for it to recreate the updated files.
- o **Matched data** is how much data the receiver got locally when recreating the updated files.

- o **File list size** is how big the file-list data was when the sender sent it to the receiver. This is smaller than the in-memory size for the file list due to some compressing of duplicated data when rsync sends the list.
- o **File list generation time** is the number of seconds that the sender spent creating the file list. This requires a modern rsync on the sending side for this to be present.
- o **File list transfer time** is the number of seconds that the sender spent sending the file list to the receiver.
- o **Total bytes sent** is the count of all the bytes that rsync sent from the client side to the server side.
- o **Total bytes received** is the count of all non-message bytes that rsync received by the client side from the server side. "Non-message" bytes means that we don't count the bytes for a verbose message that the server sent to us, which makes the stats more consistent.

--8-bit-output, -8

This tells rsync to leave all high-bit characters unescaped in the output instead of trying to test them to see if they're valid in the current locale and escaping the invalid ones. All control characters (but never tabs) are always escaped, regardless of this option's setting.

The escape idiom that started in 2.6.7 is to output a literal backslash (\) and a hash (#), followed by exactly 3 octal digits. For example, a newline would output as "\#012". A literal backslash that is in a filename is not escaped unless it is followed by a hash and 3 digits (0-9).

--human-readable, -h

Output numbers in a more human-readable format. There are 3 possible levels: (1) output numbers with a separator between each set of 3 digits (either a comma or a period, depending on if the decimal point is represented by a period or a comma); (2) output numbers in units of 1000 (with a character suffix for larger units — see below); (3) output numbers in units of 1024.

The default is human-readable level 1. Each **-h** option increases the level by one. You can take the level down to 0 (to output numbers as pure digits) by specifying the **--no-human-readable** (**--no-h**) option.

The unit letters that are appended in levels 2 and 3 are: **K** (kilo), **M** (mega), **G** (giga), **T** (tera), or **P** (peta). For example, a 1234567-byte file would output as 1.23M in level-2 (assuming that a period is your local decimal point).

Backward compatibility note: versions of rsync prior to 3.1.0 do not support human-readable level 1, and they default to level 0. Thus, specifying one or two **-h** options will behave in a comparable manner in old and new versions as long as you didn't specify a **--no-h** option prior to one or more **-h** options. See the **--list-only** option for one difference.

--partial

By default, rsync will delete any partially transferred file if the transfer is interrupted. In some circumstances it is more desirable to keep partially transferred files. Using the **--partial** option tells rsync to keep the partial file which should make a subsequent transfer of the rest of the file much faster.

--partial-dir=DIR

A better way to keep partial files than the **--partial** option is to specify a *DIR* that will be used to hold the partial data (instead of writing it out to the destination file). On the next transfer, rsync will use a file found in this dir as data to speed up the resumption of the transfer and then delete it after it has served its purpose.

Note that if **--whole-file** is specified (or implied), any partial-dir file that is found for a file that is being updated will simply be removed (since rsync is sending files without using rsync's delta-transfer algorithm).

Rsync will create the *DIR* if it is missing (just the last dir --- not the whole path). This makes it easy to use a relative path (such as "**--partial-dir=.rsync-partial**") to have rsync create the partial-directory in the destination file's directory when needed, and then remove it again when the partial file is deleted. Note that the directory is only removed if it is a relative pathname, as it is expected that an absolute path is to a directory that is reserved for partial-dir work.

If the partial-dir value is not an absolute path, rsync will add an exclude rule at the end of all your existing excludes. This will prevent the sending of any partial-dir files that may exist on the sending side, and will also prevent the untimely deletion of partial-dir items on the receiving side. An example: the above **--partial-dir** option would add the equivalent of "**-f '-p .rsync-partial/'**" at the end of any other filter rules.

If you are supplying your own exclude rules, you may need to add your own exclude/protect rule for the partial-dir because (1) the auto-added rule may be ineffective at the end of your other rules, or (2) you may wish to override rsync's exclude choice. For instance, if you want to make rsync clean-up any left-over partial-dirs that may be lying around, you should specify **--delete-after** and add a "risk" filter rule, e.g. **-f 'R .rsync-partial'**. (Avoid using **--delete-before** or **--delete-during** unless you don't need rsync to use any of the left-over partial-dir data during the current run.)

IMPORTANT: the **--partial-dir** should not be writable by other users or it is a security risk. E.g. AVOID "/tmp".

You can also set the partial-dir value the RSYNC_PARTIAL_DIR environment variable. Setting this in the environment does not force **--partial** to be enabled, but rather it affects where partial files go when **--partial** is specified. For instance, instead of using **--partial-dir=.rsync-tmp** along with **--progress**, you could set RSYNC_PARTIAL_DIR=.rsync-tmp in your environment and then just use the **-P** option to turn on the use of the .rsync-tmp dir for partial transfers. The only times that the **--partial** option does not look for this environment value are (1) when **--in-place** was specified (since **--inplace** conflicts with **--partial-dir**), and (2) when **--delay-updates** was specified (see below).

When a modern rsync resumes the transfer of a file in the partial-dir, that partial file is now updated in-place instead of creating yet another tmp-file copy (so it maxes out at dest + tmp instead of dest + partial + tmp). This requires both ends of the transfer to be at least version 3.2.0.

For the purposes of the daemon-config's "refuse options" setting, **--partial-dir** does *not* imply **--partial**. This is so that a refusal of the **--partial** option can be used to disallow the overwriting of destination files with a partial transfer, while still allowing the safer idiom provided by **--partial-dir**.

--delay-updates

This option puts the temporary file from each updated file into a holding directory until the end of the transfer, at which time all the files are renamed into place in rapid succession. This attempts to make the updating of the files a little more atomic. By default the files are placed into a directory named **.~tmp~** in each file's destination directory, but if you've specified the **--partial-dir** option, that directory will be used instead. See the comments in the **--partial-dir** section for a discussion of how this **.~tmp~** dir will be excluded from the transfer, and what you can do if you want rsync to cleanup old **.~tmp~** dirs that might be lying around. Conflicts with **--inplace** and **--append**.

This option implies **--no-inc-recursive** since it needs the full file list in memory in order to be able to iterate over it at the end.

This option uses more memory on the receiving side (one bit per file transferred) and also requires enough free disk space on the receiving side to hold an additional copy of all the updated files. Note also that you should not use an absolute path to **--partial-dir** unless (1) there is no chance of any of the files in the transfer having the same name (since all the updated files will be put into a single directory if the path is absolute) and (2) there are no mount points in the hierarchy (since the delayed updates will fail if they can't be renamed into place).

See also the "atomic-rsync" perl script in the "support" subdir for an update algorithm that is even more atomic (it uses **--link-dest** and a parallel hierarchy of files).

--prune-empty-dirs, -m

This option tells the receiving rsync to get rid of empty directories from the file-list, including nested directories that have no non-directory children. This is useful for avoiding the creation of a bunch of useless directories when the sending rsync is recursively scanning a hierarchy of files using include/exclude/filter rules.

Note that the use of transfer rules, such as the **--min-size** option, does not affect what goes into the file list, and thus does not leave directories empty, even if none of the files in a directory match the transfer rule.

Because the file-list is actually being pruned, this option also affects what directories get deleted when a delete is active. However, keep in mind that excluded files and directories can prevent existing items from being deleted due to an exclude both hiding source files and protecting destination files. See the perishable filter-rule option for how to avoid this.

You can prevent the pruning of certain empty directories from the file-list by using a global "protect" filter. For instance, this option would ensure that the directory "emptydir" was kept in the file-list:

```
--filter 'protect emptydir/'
```

Here's an example that copies all .pdf files in a hierarchy, only creating the necessary destination directories to hold the .pdf files, and ensures that any superfluous files and directories in the destination are removed (note the hide filter of non-directories being used instead of an exclude):

```
rsync -avm --del --include='*.pdf' -f 'hide,! /*' src/ dest
```

If you didn't want to remove superfluous destination files, the more time-honored options of **--include='/*'** **--exclude='/*'** would work fine in place of the hide-filter (if that is more natural to you).

--progress

This option tells rsync to print information showing the progress of the transfer. This gives a bored user something to watch. With a modern rsync this is the same as specifying **--info=flist2,name,progress**, but any user-supplied settings for those info flags takes precedence (e.g. "**--info=flist0 --progress**").

While rsync is transferring a regular file, it updates a progress line that looks like this:

```
782448 63% 110.64kB/s 0:00:04
```

In this example, the receiver has reconstructed 782448 bytes or 63% of the sender's file, which is being reconstructed at a rate of 110.64 kilobytes per second, and the transfer will finish in 4 seconds if the current rate is maintained until the end.

These statistics can be misleading if rsync's delta-transfer algorithm is in use. For example, if the sender's file consists of the basis file followed by additional data, the reported rate will probably drop dramatically when the receiver gets to the literal data, and the transfer will probably take much longer to finish than the receiver estimated as it was finishing the matched part of the file.

When the file transfer finishes, rsync replaces the progress line with a summary line that looks like this:

```
1,238,099 100% 146.38kB/s 0:00:08 (xfr#5, to-chk=169/396)
```

In this example, the file was 1,238,099 bytes long in total, the average rate of transfer for the whole file was 146.38 kilobytes per second over the 8 seconds that it took to complete, it was the 5th transfer of a regular file during the current rsync session, and there are 169 more files for the receiver to check (to see if they are up-to-date or not) remaining out of the 396 total files in the file-list.

In an incremental recursion scan, rsync won't know the total number of files in the file-list until it reaches the ends of the scan, but since it starts to transfer files during the scan, it will display a line with the text "ir-chk" (for incremental recursion check) instead of "to-chk" until the point that it knows the full size of the list, at which point it will switch to using "to-chk". Thus, seeing "ir-chk" lets you know that the total count of files in the file list is still going to increase (and each time it does, the count of files left to check will increase by the number of the files added to the list).

- P** The **-P** option is equivalent to **--partial --progress**. Its purpose is to make it much easier to specify these two options for a long transfer that may be interrupted.

There is also a **--info=progress2** option that outputs statistics based on the whole transfer, rather than individual files. Use this flag without outputting a filename (e.g. avoid **-v** or specify **--info=name0**) if you want to see how the transfer is doing without scrolling the screen with a lot of names. (You don't need to specify the **--progress** option in order to use **--info=progress2**.)

Finally, you can get an instant progress report by sending rsync a signal of either SIGINFO or SIGVTALRM. On BSD systems, a SIGINFO is generated by typing a Ctrl+T (Linux doesn't currently support a SIGINFO signal). When the client-side process receives one of those signals, it sets a flag to output a single progress report which is output when the current file transfer finishes (so it may take a little time if a big file is being handled when the signal arrives). A filename is output (if needed) followed by the **--info=progress2** format of progress info. If you don't know which of the 3 rsync processes is the client process, it's OK to signal all of them (since the non-client processes ignore the signal).

CAUTION: sending SIGVTALRM to an older rsync (pre-3.2.0) will kill it.

--password-file=FILE

This option allows you to provide a password for accessing an rsync daemon via a file or via standard input if **FILE** is **-**. The file should contain just the password on the first line (all other lines are ignored). Rsync will exit with an error if **FILE** is world readable or if a root-run rsync command finds a non-root-owned file.

This option does not supply a password to a remote shell transport such as ssh; to learn how to do that, consult the remote shell's documentation. When accessing an rsync daemon using a remote shell as the transport, this option only comes into effect after the remote shell finishes its authentication (i.e. if you have also specified a password in the daemon's config file).

--early-input=FILE

This option allows rsync to send up to 5K of data to the "early exec" script on its stdin. One possible use of this data is to give the script a secret that can be used to mount an encrypted filesystem (which you should unmount in the the "post-xfer exec" script).

The daemon must be at least version 3.2.1.

--list-only

This option will cause the source files to be listed instead of transferred. This option is inferred if there is a single source arg and no destination specified, so its main uses are: (1) to turn a copy command that includes a destination arg into a file-listing command, or (2) to be able to specify more than one source arg (note: be sure to include the destination). Caution: keep in mind that a source arg with a wild-card is expanded by the shell into multiple args, so it is never safe to try to list such an arg without using this option. For example:

```
rsync -av --list-only foo* dest/
```

Starting with rsync 3.1.0, the sizes output by **--list-only** are affected by the **--human-readable** option. By default they will contain digit separators, but higher levels of readability will output the sizes with unit suffixes. Note also that the column width for the size output has increased from 11 to 14 characters for all human-readable levels. Use **--no-h** if you want just digits in the sizes, and the old column width of 11 characters.

Compatibility note: when requesting a remote listing of files from an rsync that is version 2.6.3 or older, you may encounter an error if you ask for a non-recursive listing. This is because a file

listing implies the **--dirs** option w/o **--recursive**, and older rsyncs don't have that option. To avoid this problem, either specify the **--no-dirs** option (if you don't need to expand a directory's content), or turn on recursion and exclude the content of subdirectories: **-r --exclude='/*/*'**.

--bwlimit=RATE

This option allows you to specify the maximum transfer rate for the data sent over the socket, specified in units per second. The RATE value can be suffixed with a string to indicate a size multiplier, and may be a fractional value (e.g. "**--bwlimit=1.5m**"). If no suffix is specified, the value will be assumed to be in units of 1024 bytes (as if "K" or "KiB" had been appended). See the **--max-size** option for a description of all the available suffixes. A value of 0 specifies no limit.

For backward-compatibility reasons, the rate limit will be rounded to the nearest KiB unit, so no rate smaller than 1024 bytes per second is possible.

Rsync writes data over the socket in blocks, and this option both limits the size of the blocks that rsync writes, and tries to keep the average transfer rate at the requested limit. Some burstiness may be seen where rsync writes out a block of data and then sleeps to bring the average rate into compliance.

Due to the internal buffering of data, the **--progress** option may not be an accurate reflection on how fast the data is being sent. This is because some files can show up as being rapidly sent when the data is quickly buffered, while others can show up as very slow when the flushing of the output buffer occurs. This may be fixed in a future version.

'**--stop-after=MINS**

This option tells rsync to stop copying when the specified number of minutes has elapsed.

Rsync also accepts an earlier version of this option: **--time-limit=MINS**.

For maximal flexibility, rsync does not communicate this option to the remote rsync since it is usually enough that one side of the connection quits as specified. This allows the option's use even when only one side of the connection supports it. You can tell the remote side about the time limit using **--remote-option (-M)**, should the need arise.

'**--stop-at=y-m-dTh:m**

This option tells rsync to stop copying when the specified point in time has been reached. The date & time can be fully specified in a numeric format of year-month-dayThour:minute (e.g. 2000-12-31T23:59) in the local timezone. You may choose to separate the date numbers using slashes instead of dashes.

The value can also be abbreviated in a variety of ways, such as specifying a 2-digit year and/or leaving off various values. In all cases, the value will be taken to be the next possible point in time where the supplied information matches. If the value specifies the current time or a past time, rsync exits with an error.

For example, "1-30" specifies the next January 30th (at midnight local time), "14:00" specifies the next 2 P.M., "1" specifies the next 1st of the month at midnight, "31" specifies the next month where we can stop on its 31st day, and ":59" specifies the next 59th minute after the hour.

For maximal flexibility, rsync does not communicate this option to the remote rsync since it is usually enough that one side of the connection quits as specified. This allows the option's use even when only one side of the connection supports it. You can tell the remote side about the time limit using **--remote-option (-M)**, should the need arise. Do keep in mind that the remote host may have a different default timezone than your local host.

--write-batch=FILE

Record a file that can later be applied to another identical destination with **--read-batch**. See the "BATCH MODE" section for details, and also the **--only-write-batch** option.

This option overrides the negotiated checksum & compress lists and always negotiates a choice based on old-school md5/md4/zlib choices. If you want a more modern choice, use the **--checksum-choice (--cc)** and/or **--compress-choice (--zc)** options.

--only-write-batch=FILE

Works like **--write-batch**, except that no updates are made on the destination system when creating the batch. This lets you transport the changes to the destination system via some other means and then apply the changes via **--read-batch**.

Note that you can feel free to write the batch directly to some portable media: if this media fills to capacity before the end of the transfer, you can just apply that partial transfer to the destination and repeat the whole process to get the rest of the changes (as long as you don't mind a partially updated destination system while the multi-update cycle is happening).

Also note that you only save bandwidth when pushing changes to a remote system because this allows the batched data to be diverted from the sender into the batch file without having to flow over the wire to the receiver (when pulling, the sender is remote, and thus can't write the batch).

--read-batch=FILE

Apply all of the changes stored in FILE, a file previously generated by **--write-batch**. If FILE is `-`, the batch data will be read from standard input. See the "BATCH MODE" section for details.

--protocol=NUM

Force an older protocol version to be used. This is useful for creating a batch file that is compatible with an older version of rsync. For instance, if rsync 2.6.4 is being used with the **--write-batch** option, but rsync 2.6.3 is what will be used to run the **--read-batch** option, you should use "`--protocol=28`" when creating the batch file to force the older protocol version to be used in the batch file (assuming you can't upgrade the rsync on the reading system).

--iconv=CONVERT_SPEC

Rsync can convert filenames between character sets using this option. Using a CONVERT_SPEC of `".."` tells rsync to look up the default character-set via the locale setting. Alternately, you can fully specify what conversion to do by giving a local and a remote charset separated by a comma in the order **--iconv=LOCAL,REMOTE**, e.g. **--iconv=utf8,iso88591**. This order ensures that the option will stay the same whether you're pushing or pulling files. Finally, you can specify either **--no-iconv** or a CONVERT_SPEC of `"-"` to turn off any conversion. The default setting of this option is site-specific, and can also be affected via the RSYNC_ICONV environment variable.

For a list of what charset names your local iconv library supports, you can run `"iconv --list"`.

If you specify the **--protect-args** option (`-s`), rsync will translate the filenames you specify on the command-line that are being sent to the remote host. See also the **--files-from** option.

Note that rsync does not do any conversion of names in filter files (including include/exclude files). It is up to you to ensure that you're specifying matching rules that can match on both sides of the transfer. For instance, you can specify extra include/exclude rules if there are filename differences on the two sides that need to be accounted for.

When you pass an **--iconv** option to an rsync daemon that allows it, the daemon uses the charset specified in its "charset" configuration parameter regardless of the remote charset you actually pass. Thus, you may feel free to specify just the local charset for a daemon transfer (e.g. **--iconv=utf8**).

--ipv4, -4 or --ipv6, -6

Tells rsync to prefer IPv4/IPv6 when creating sockets or running ssh. This affects sockets that rsync has direct control over, such as the outgoing socket when directly contacting an rsync daemon, as well as the forwarding of the **-4** or **-6** option to ssh when rsync can deduce that ssh is being used as the remote shell. For other remote shells you'll need to specify the "**--rsh SHELL -4**" option directly (or whatever ipv4/ipv6 hint options it uses).

These options also exist in the **--daemon** mode section.

If rsync was compiled without support for IPv6, the **--ipv6** option will have no effect. The **rsync --version** output will contain **"no IPv6"** if is the case.

--checksum-seed=NUM

Set the checksum seed to the integer NUM. This 4 byte checksum seed is included in each block and MD4 file checksum calculation (the more modern MD5 file checksums don't use a seed). By default the checksum seed is generated by the server and defaults to the current **time()**. This option is used to set a specific checksum seed, which is useful for applications that want repeatable block checksums, or in the case where the user wants a more random checksum seed. Setting NUM to 0 causes rsync to use the default of **time()** for checksum seed.

DAEMON OPTIONS

The options allowed when starting an rsync daemon are as follows:

--daemon

This tells rsync that it is to run as a daemon. The daemon you start running may be accessed using an rsync client using the **host::module** or **rsync://host/module/** syntax.

If standard input is a socket then rsync will assume that it is being run via inetd, otherwise it will detach from the current terminal and become a background daemon. The daemon will read the config file (**rsyncd.conf**) on each connect made by a client and respond to requests accordingly. See the **rsyncd.conf(5)** man page for more details.

--address=ADDRESS

By default rsync will bind to the wildcard address when run as a daemon with the **--daemon** option. The **--address** option allows you to specify a specific IP address (or hostname) to bind to. This makes virtual hosting possible in conjunction with the **--config** option. See also the "address" global option in the **rsyncd.conf** manpage.

--bwlimit=RATE

This option allows you to specify the maximum transfer rate for the data the daemon sends over the socket. The client can still specify a smaller **--bwlimit** value, but no larger value will be allowed. See the client version of this option (above) for some extra details.

--config=FILE

This specifies an alternate config file than the default. This is only relevant when **--daemon** is specified. The default is **/etc/rsyncd.conf** unless the daemon is running over a remote shell program and the remote user is not the super-user; in that case the default is **rsyncd.conf** in the current directory (typically **\$HOME**).

--dparam=OVERRIDE, -M

This option can be used to set a daemon-config parameter when starting up rsync in daemon mode. It is equivalent to adding the parameter at the end of the global settings prior to the first module's definition. The parameter names can be specified without spaces, if you so desire. For instance:

```
rsync --daemon -M pidfile=/path/rsync.pid
```

--no-detach

When running as a daemon, this option instructs rsync to not detach itself and become a background process. This option is required when running as a service on Cygwin, and may also be useful when rsync is supervised by a program such as **daemontools** or AIX's **System Resource Controller**. **--no-detach** is also recommended when rsync is run under a debugger. This option has no effect if rsync is run from inetd or sshd.

--port=PORT

This specifies an alternate TCP port number for the daemon to listen on rather than the default of 873. See also the "port" global option in the **rsyncd.conf** manpage.

--log-file=FILE

This option tells the rsync daemon to use the given log-file name instead of using the "**log file**" setting in the config file.

--log-file-format=FORMAT

This option tells the rsync daemon to use the given FORMAT string instead of using the "**log format**" setting in the config file. It also enables "**transfer logging**" unless the string is empty, in

which case transfer logging is turned off.

--sockopts

This overrides the **socket options** setting in the rsyncd.conf file and has the same syntax.

--verbose, -v

This option increases the amount of information the daemon logs during its startup phase. After the client connects, the daemon's verbosity level will be controlled by the options that the client used and the "**max verbosity**" setting in the module's config section.

--ipv4, -4 or --ipv6, -6

Tells rsync to prefer IPv4/IPv6 when creating the incoming sockets that the rsync daemon will use to listen for connections. One of these options may be required in older versions of Linux to work around an IPv6 bug in the kernel (if you see an "address already in use" error when nothing else is using the port, try specifying **--ipv6** or **--ipv4** when starting the daemon).

These options also exist in the regular rsync options section.

If rsync was compiled without support for IPv6, the **--ipv6** option will have no effect. The **rsync --version** output will contain "**no IPv6**" if is the case.

--help, -h

When specified after **--daemon**, print a short help page describing the options available for starting an rsync daemon.

FILTER RULES

The filter rules allow for flexible selection of which files to transfer (include) and which files to skip (exclude). The rules either directly specify include/exclude patterns or they specify a way to acquire more include/exclude patterns (e.g. to read them from a file).

As the list of files/directories to transfer is built, rsync checks each name to be transferred against the list of include/exclude patterns in turn, and the first matching pattern is acted on: if it is an exclude pattern, then that file is skipped; if it is an include pattern then that filename is not skipped; if no matching pattern is found, then the filename is not skipped.

Rsync builds an ordered list of filter rules as specified on the command-line. Filter rules have the following syntax:

```
RULE [PATTERN_OR_FILENAME]
      RULE,MODIFIERS [PATTERN_OR_FILENAME]
```

You have your choice of using either short or long RULE names, as described below. If you use a short-named rule, the ',' separating the RULE from the MODIFIERS is optional. The PATTERN or FILENAME that follows (when present) must come after either a single space or an underscore (_). Here are the available rule prefixes:

exclude, '-'

specifies an exclude pattern.

include, '+'

specifies an include pattern.

merge, ':'

specifies a merge-file to read for more rules.

dir-merge, ':'

specifies a per-directory merge-file.

hide, 'H'

specifies a pattern for hiding files from the transfer.

show, 'S'

files that match the pattern are not hidden.

protect, 'P'

specifies a pattern for protecting files from deletion.

risk, 'R'

files that match the pattern are not protected.

clear, '!'*

clears the current include/exclude list (takes no arg)

When rules are being read from a file, empty lines are ignored, as are whole-line comments that start with a '#' (filename rules that contain a hash are unaffected).

Note that the **--include** & **--exclude** command-line options do not allow the full range of rule parsing as described above — they only allow the specification of include / exclude patterns plus a "!" token to clear the list (and the normal comment parsing when rules are read from a file). If a pattern does not begin with "-" (dash, space) or "+" (plus, space), then the rule will be interpreted as if "+" (for an include option) or "-" (for an exclude option) were prefixed to the string. A **--filter** option, on the other hand, must always contain either a short or long rule name at the start of the rule.

Note also that the **--filter**, **--include**, and **--exclude** options take one rule/pattern each. To add multiple ones, you can repeat the options on the command-line, use the merge-file syntax of the **--filter** option, or the **--include-from** / **--exclude-from** options.

INCLUDE/EXCLUDE PATTERN RULES

You can include and exclude files by specifying patterns using the "+", "-", etc. filter rules (as introduced in the FILTER RULES section above). The include/exclude rules each specify a pattern that is matched against the names of the files that are going to be transferred. These patterns can take several forms:

- o if the pattern starts with a / then it is anchored to a particular spot in the hierarchy of files, otherwise it is matched against the end of the pathname. This is similar to a leading ^ in regular expressions. Thus/**f oo** would match a name of "foo" at either the "root of the transfer" (for a global rule) or in the merge-file's directory (for a per-directory rule). An unqualified **foo** would match a name of "foo" anywhere in the tree because the algorithm is applied recursively from the top down; it behaves as if each path component gets a turn at being the end of the filename. Even the unanchored "sub/foo" would match at any point in the hierarchy where a "foo" was found within a directory named "sub". See the section on ANCHORING INCLUDE/EXCLUDE PATTERNS for a full discussion of how to specify a pattern that matches at the root of the transfer.
- o if the pattern ends with a / then it will only match a directory, not a regular file, symlink, or device.
- o rsync chooses between doing a simple string match and wildcard matching by checking if the pattern contains one of these three wildcard characters: '*', '?', and '[' .
- o a '*' matches any path component, but it stops at slashes.
- o use '**' to match anything, including slashes.
- o a '?' matches any character except a slash (/).
- o a '[' introduces a character class, such as **[a-z]** or **[:alpha:]**.
- o in a wildcard pattern, a backslash can be used to escape a wildcard character, but it is matched literally when no wildcards are present. This means that there is an extra level of backslash removal when a pattern contains wildcard characters compared to a pattern that has none. e.g. if you add a wildcard to "**foo\bar**" (which matches the backslash) you would need to use "**foo\\bar***" to avoid the "\b" becoming just "b".
- o if the pattern contains a / (not counting a trailing /) or a "**", then it is matched against the full pathname, including any leading directories. If the pattern doesn't contain a / or a "**", then it is matched only against the final component of the filename. (Remember that the algorithm is applied recursively so "full filename" can actually be any portion of a path from the starting directory on down.)

- o a trailing "**dir_name**/**" will match both the directory (as if "dir_name/" had been specified) and everything in the directory (as if "dir_name/*" had been specified). This behavior was added in version 2.6.7.

Note that, when using the **--recursive (-r)** option (which is implied by **-a**), every subdir component of every path is visited left to right, with each directory having a chance for exclusion before its content. In this way include/exclude patterns are applied recursively to the pathname of each node in the filesystem's tree (those inside the transfer). The exclude patterns short-circuit the directory traversal stage as rsync finds the files to send.

For instance, to include "**/foo/bar/baz**", the directories "**/foo**" and "**/foo/bar**" must not be excluded. Excluding one of those parent directories prevents the examination of its content, cutting off rsync's recursion into those paths and rendering the include for "**/foo/bar/baz**" ineffectual (since rsync can't match something it never sees in the cut-off section of the directory hierarchy).

The concept path exclusion is particularly important when using a trailing '*' rule. For instance, this won't work:

```
+ /some/path/this-file-will-not-be-found
+ /file-is-included
- *
```

This fails because the parent directory "some" is excluded by the '*' rule, so rsync never visits any of the files in the "some" or "some/path" directories. One solution is to ask for all directories in the hierarchy to be included by using a single rule: "+ /*" (put it somewhere before the "- *" rule), and perhaps use the **--prune-empty-dirs** option. Another solution is to add specific include rules for all the parent dirs that need to be visited. For instance, this set of rules works fine:

```
+ /some/
+ /some/path/
+ /some/path/this-file-is-found
+ /file-also-included
- *
```

Here are some examples of exclude/include matching:

- o "**- *.o**" would exclude all names matching ***.o**
- o "**- /foo**" would exclude a file (or directory) named foo in the transfer-root directory
- o "**- foo/**" would exclude any directory named foo
- o "**- /foo/*/bar**" would exclude any file named bar which is at two levels below a directory named foo in the transfer-root directory
- o "**- /foo/**/bar**" would exclude any file named bar two or more levels below a directory named foo in the transfer-root directory
- o The combination of "+ /*", "+ *.c", and "- *" would include all directories and C source files but nothing else (see also the **--prune-empty-dirs** option)
- o The combination of "**+ foo/**", "**+ foo/bar.c**", and "**- ***" would include only the foo directory and foo/bar.c (the foo directory must be explicitly included or it would be excluded by the "*")

The following modifiers are accepted after a "+" or "-":

- o A **/** specifies that the include/exclude rule should be matched against the absolute pathname of the current item. For example, "**-/ /etc/passwd**" would exclude the passwd file any time the transfer was sending files from the "/etc" directory, and "**-/ subdir/foo**" would always exclude "foo" when it is in a dir named "subdir", even if "foo" is at the root of the current transfer.
- o A **!** specifies that the include/exclude should take effect if the pattern fails to match. For instance, "**-! /***" would exclude all non-directories.

- o A **C** is used to indicate that all the global CVS-exclude rules should be inserted as excludes in place of the "-C". No arg should follow.
- o An **s** is used to indicate that the rule applies to the sending side. When a rule affects the sending side, it prevents files from being transferred. The default is for a rule to affect both sides unless **--delete-excluded** was specified, in which case default rules become sender-side only. See also the hide (H) and show (S) rules, which are an alternate way to specify sending-side includes/excludes.
- o An **r** is used to indicate that the rule applies to the receiving side. When a rule affects the receiving side, it prevents files from being deleted. See the **s** modifier for more info. See also the protect (P) and risk (R) rules, which are an alternate way to specify receiver-side includes/excludes.
- o A **p** indicates that a rule is perishable, meaning that it is ignored in directories that are being deleted. For instance, the **-C** option's default rules that exclude things like "CVS" and "*.**o**" are marked as perishable, and will not prevent a directory that was removed on the source from being deleted on the destination.
- o An **x** indicates that a rule affects xattr names in xattr copy/delete operations (and is thus ignored when matching file/dir names). If no xattr-matching rules are specified, a default xattr filtering rule is used (see the **--xattrs** option).

MERGE-FILE FILTER RULES

You can merge whole files into your filter rules by specifying either a merge (.) or a dir-merge (:) filter rule (as introduced in the FILTER RULES section above).

There are two kinds of merged files -- single-instance ('.') and per-directory (':'). A single-instance merge file is read one time, and its rules are incorporated into the filter list in the place of the "." rule. For per-directory merge files, rsync will scan every directory that it traverses for the named file, merging its contents when the file exists into the current list of inherited rules. These per-directory rule files must be created on the sending side because it is the sending side that is being scanned for the available files to transfer. These rule files may also need to be transferred to the receiving side if you want them to affect what files don't get deleted (see PER-DIRECTORY RULES AND DELETE below).

Some examples:

```
merge /etc/rsync/default.rules
./etc/rsync/default.rules
dir-merge .per-dir-filter
dir-merge,n- .non-inherited-per-dir-excludes
:n- .non-inherited-per-dir-excludes
```

The following modifiers are accepted after a merge or dir-merge rule:

- o A **-** specifies that the file should consist of only exclude patterns, with no other rule-parsing except for in-file comments.
- o A **+** specifies that the file should consist of only include patterns, with no other rule-parsing except for in-file comments.
- o A **C** is a way to specify that the file should be read in a CVS-compatible manner. This turns on '**n**', '**w**', and '**-**', but also allows the list-clearing token (!) to be specified. If no filename is provided, ".cvignore" is assumed.
- o A **e** will exclude the merge-file name from the transfer; e.g. "dir-merge,e .rules" is like "dir-merge .rules" and "- .rules".
- o An **n** specifies that the rules are not inherited by subdirectories.
- o A **w** specifies that the rules are word-split on whitespace instead of the normal line-splitting. This also turns off comments. Note: the space that separates the prefix from the rule is treated specially, so "**- foo + bar**" is parsed as two rules (assuming that prefix-parsing wasn't also disabled).

- o You may also specify any of the modifiers for the "+" or "-" rules (above) in order to have the rules that are read in from the file default to having that modifier set (except for the ! modifier, which would not be useful). For instance, "merge,-/.excl" would treat the contents of .excl as absolute-path excludes, while "dir-merge,s .filt" and ":sC" would each make all their per-directory rules apply only on the sending side. If the merge rule specifies sides to affect (via the **s** or **r** modifier or both), then the rules in the file must not specify sides (via a modifier or a rule prefix such as **hide**).

Per-directory rules are inherited in all subdirectories of the directory where the merge-file was found unless the '**n**' modifier was used. Each subdirectory's rules are prefixed to the inherited per-directory rules from its parents, which gives the newest rules a higher priority than the inherited rules. The entire set of dir-merge rules are grouped together in the spot where the merge-file was specified, so it is possible to override dir-merge rules via a rule that got specified earlier in the list of global rules. When the list-clearing rule ("!") is read from a per-directory file, it only clears the inherited rules for the current merge file.

Another way to prevent a single rule from a dir-merge file from being inherited is to anchor it with a leading slash. Anchored rules in a per-directory merge-file are relative to the merge-file's directory, so a pattern "/foo" would only match the file "foo" in the directory where the dir-merge filter file was found.

Here's an example filter file which you'd specify via **--filter=". file"**:

```
merge /home/user/.global-filter
- *.gz
dir-merge .rules
+ *.[ch]
- *.o
- foo*
```

This will merge the contents of the /home/user/.global-filter file at the start of the list and also turns the ".rules" filename into a per-directory filter file. All rules read in prior to the start of the directory scan follow the global anchoring rules (i.e. a leading slash matches at the root of the transfer).

If a per-directory merge-file is specified with a path that is a parent directory of the first transfer directory, rsync will scan all the parent dirs from that starting point to the transfer directory for the indicated per-directory file. For instance, here is a common filter (see **-F**):

```
--filter=':./rsync-filter'
```

That rule tells rsync to scan for the file .rsync-filter in all directories from the root down through the parent directory of the transfer prior to the start of the normal directory scan of the file in the directories that are sent as a part of the transfer. (Note: for an rsync daemon, the root is always the same as the module's "path".)

Some examples of this pre-scanning for per-directory files:

```
rsync -avF /src/path/ /dest/dir
rsync -av --filter=': ../../.rsync-filter' /src/path/ /dest/dir
rsync -av --filter=': .rsync-filter' /src/path/ /dest/dir
```

The first two commands above will look for ".rsync-filter" in "/" and "/src" before the normal scan begins looking for the file in "/src/path" and its subdirectories. The last command avoids the parent-dir scan and only looks for the ".rsync-filter" files in each directory that is a part of the transfer.

If you want to include the contents of a ".cvsignore" in your patterns, you should use the rule ":C", which creates a dir-merge of the .cvsignore file, but parsed in a CVS-compatible manner. You can use this to affect where the **--cvs-exclude (-C)** option's inclusion of the per-directory .cvsignore file gets placed into your rules by putting the ":C" wherever you like in your filter rules. Without this, rsync would add the dir-merge rule for the .cvsignore file at the end of all your other rules (giving it a lower priority than your command-line rules). For example:

```
cat <<EOT | rsync -avC --filter=': -' a/ b
+ foo.o
```

```
:C
- *.old
EOT
rsync -avC --include=foo.o -f :C --exclude='*.old' a/ b
```

Both of the above rsync commands are identical. Each one will merge all the per-directory .cvsignore rules in the middle of the list rather than at the end. This allows their dir-specific rules to supersede the rules that follow the :C instead of being subservient to all your rules. To affect the other CVS exclude rules (i.e. the default list of exclusions, the contents of \$HOME/.cvsignore, and the value of \$CVSIGNORE) you should omit the **-C** command-line option and instead insert a "**-C**" rule into your filter rules; e.g. "**--filter=-C**".

LIST-CLEARING FILTER RULE

You can clear the current include/exclude list by using the "!" filter rule (as introduced in the FILTER RULES section above). The "current" list is either the global list of rules (if the rule is encountered while parsing the filter options) or a set of per-directory rules (which are inherited in their own sub-list, so a sub-directory can use this to clear out the parent's rules).

ANCHORING INCLUDE/EXCLUDE PATTERNS

As mentioned earlier, global include/exclude patterns are anchored at the "root of the transfer" (as opposed to per-directory patterns, which are anchored at the merge-file's directory). If you think of the transfer as a subtree of names that are being sent from sender to receiver, the transfer-root is where the tree starts to be duplicated in the destination directory. This root governs where patterns that start with a / match.

Because the matching is relative to the transfer-root, changing the trailing slash on a source path or changing your use of the **--relative** option affects the path you need to use in your matching (in addition to changing how much of the file tree is duplicated on the destination host). The following examples demonstrate this.

Let's say that we want to match two source files, one with an absolute path of "/home/me/foo/bar", and one with a path of "/home/you/bar/baz". Here is how the various command choices differ for a 2-source transfer:

Example cmd: rsync -a /home/me /home/you /dest
 +/- pattern: /me/foo/bar
 +/- pattern: /you/bar/baz
 Target file: /dest/me/foo/bar
 Target file: /dest/you/bar/baz

Example cmd: rsync -a /home/me/ /home/you/ /dest
 +/- pattern: /foo/bar (note missing "me")
 +/- pattern: /bar/baz (note missing "you")
 Target file: /dest/foo/bar
 Target file: /dest/bar/baz

Example cmd: rsync -a --relative /home/me/ /home/you /dest
 +/- pattern: /home/me/foo/bar (note full path)
 +/- pattern: /home/you/bar/baz (ditto)
 Target file: /dest/home/me/foo/bar
 Target file: /dest/home/you/bar/baz

Example cmd: cd /home; rsync -a --relative me/foo you/ /dest
 +/- pattern: /me/foo/bar (starts at specified path)
 +/- pattern: /you/bar/baz (ditto)
 Target file: /dest/me/foo/bar
 Target file: /dest/you/bar/baz

The easiest way to see what name you should filter is to just look at the output when using **--verbose** and put a / in front of the name (use the **--dry-run** option if you're not yet ready to copy any files).

PER-DIRECTORY RULES AND DELETE

Without a delete option, per-directory rules are only relevant on the sending side, so you can feel free to exclude the merge files themselves without affecting the transfer. To make this easy, the '`e`' modifier adds this exclude for you, as seen in these two equivalent commands:

```
rsync -av --filter=': .excl' --exclude=.excl host:src/dir /dest
rsync -av --filter=':e .excl' host:src/dir /dest
```

However, if you want to do a delete on the receiving side AND you want some files to be excluded from being deleted, you'll need to be sure that the receiving side knows what files to exclude. The easiest way is to include the per-directory merge files in the transfer and use **--delete-after**, because this ensures that the receiving side gets all the same exclude rules as the sending side before it tries to delete anything:

```
rsync -avF --delete-after host:src/dir /dest
```

However, if the merge files are not a part of the transfer, you'll need to either specify some global exclude rules (i.e. specified on the command line), or you'll need to maintain your own per-directory merge files on the receiving side. An example of the first is this (assume that the remote `.rules` files exclude themselves):

```
rsync -av --filter=': .rules' --filter=': ./my/extra.rules'
--delete host:src/dir /dest
```

In the above example the `extra.rules` file can affect both sides of the transfer, but (on the sending side) the rules are subservient to the rules merged from the `.rules` files because they were specified after the per-directory merge rule.

In one final example, the remote side is excluding the `.rsync-filter` files from the transfer, but we want to use our own `.rsync-filter` files to control what gets deleted on the receiving side. To do this we must specifically exclude the per-directory merge files (so that they don't get deleted) and then put rules into the local files to control what else should not get deleted. Like one of these commands:

```
rsync -av --filter=':e /.rsync-filter' --delete \
host:src/dir /dest
rsync -avFF --delete host:src/dir /dest
```

BATCH MODE

Batch mode can be used to apply the same set of updates to many identical systems. Suppose one has a tree which is replicated on a number of hosts. Now suppose some changes have been made to this source tree and those changes need to be propagated to the other hosts. In order to do this using batch mode, rsync is run with the write-batch option to apply the changes made to the source tree to one of the destination trees. The write-batch option causes the rsync client to store in a "batch file" all the information needed to repeat this operation against other, identical destination trees.

Generating the batch file once saves having to perform the file status, checksum, and data block generation more than once when updating multiple destination trees. Multicast transport protocols can be used to transfer the batch update files in parallel to many hosts at once, instead of sending the same data to every host individually.

To apply the recorded changes to another destination tree, run rsync with the read-batch option, specifying the name of the same batch file, and the destination tree. Rsync updates the destination tree using the information stored in the batch file.

For your convenience, a script file is also created when the write-batch option is used: it will be named the same as the batch file with ".sh" appended. This script file contains a command-line suitable for updating a destination tree using the associated batch file. It can be executed using a Bourne (or Bourne-like) shell, optionally passing in an alternate destination tree pathname which is then used instead of the original destination path. This is useful when the destination tree path on the current host differs from the one used to create the batch file.

Examples:

```
$ rsync --write-batch=foo -a host:/source/dir/ /adest/dir/
$ scp foo* remote:
```

```
$ ssh remote ./foo.sh /bdest/dir/
$ rsync --write-batch=foo -a /source/dir/ /adest/dir/
$ ssh remote rsync --read-batch=- -a /bdest/dir/ <foo
```

In these examples, rsync is used to update /adest/dir/ from /source/dir/ and the information to repeat this operation is stored in "foo" and "foo.sh". The host "remote" is then updated with the batched data going into the directory /bdest/dir. The differences between the two examples reveals some of the flexibility you have in how you deal with batches:

- o The first example shows that the initial copy doesn't have to be local -- you can push or pull data to/from a remote host using either the remote-shell syntax or rsync daemon syntax, as desired.
- o The first example uses the created "foo.sh" file to get the right rsync options when running the read-batch command on the remote host.
- o The second example reads the batch data via standard input so that the batch file doesn't need to be copied to the remote machine first. This example avoids the foo.sh script because it needed to use a modified **--read-batch** option, but you could edit the script file if you wished to make use of it (just be sure that no other option is trying to use standard input, such as the "**--exclude-from=-**" option).

Caveats:

The read-batch option expects the destination tree that it is updating to be identical to the destination tree that was used to create the batch update fileset. When a difference between the destination trees is encountered the update might be discarded with a warning (if the file appears to be up-to-date already) or the file-update may be attempted and then, if the file fails to verify, the update discarded with an error. This means that it should be safe to re-run a read-batch operation if the command got interrupted. If you wish to force the batched-update to always be attempted regardless of the file's size and date, use the **-I** option (when reading the batch). If an error occurs, the destination tree will probably be in a partially updated state. In that case, rsync can be used in its regular (non-batch) mode of operation to fix up the destination tree.

The rsync version used on all destinations must be at least as new as the one used to generate the batch file. Rsync will die with an error if the protocol version in the batch file is too new for the batch-reading rsync to handle. See also the **--proto** option for a way to have the creating rsync generate a batch file that an older rsync can understand. (Note that batch files changed format in version 2.6.3, so mixing versions older than that with newer versions will not work.)

When reading a batch file, rsync will force the value of certain options to match the data in the batch file if you didn't set them to the same as the batch-writing command. Other options can (and should) be changed. For instance **--write-batch** changes to **--read-batch**, **--files-from** is dropped, and the **--filter** / **--include** / **--exclude** options are not needed unless one of the **--delete** options is specified.

The code that creates the BATCH.sh file transforms any filter/include/exclude options into a single list that is appended as a "here" document to the shell script file. An advanced user can use this to modify the exclude list if a change in what gets deleted by **--delete** is desired. A normal user can ignore this detail and just use the shell script as an easy way to run the appropriate **--read-batch** command for the batched data.

The original batch mode in rsync was based on "rsync+", but the latest version uses a new implementation.

SYMBOLIC LINKS

Three basic behaviors are possible when rsync encounters a symbolic link in the source directory.

By default, symbolic links are not transferred at all. A message "skipping non-regular" file is emitted for any symlinks that exist.

If **--links** is specified, then symlinks are recreated with the same target on the destination. Note that **--archive** implies **--links**.

If **--copy-links** is specified, then symlinks are "collapsed" by copying their referent, rather than the symlink.

Rsync can also distinguish "safe" and "unsafe" symbolic links. An example where this might be used is a

web site mirror that wishes to ensure that the rsync module that is copied does not include symbolic links to **/etc/passwd** in the public section of the site. Using **--copy-unsafe-links** will cause any links to be copied as the file they point to on the destination. Using **--safe-links** will cause unsafe links to be omitted altogether. (Note that you must specify **--links** for **--safe-links** to have any effect.)

Symbolic links are considered unsafe if they are absolute symlinks (start with `/`), empty, or if they contain enough `..` components to ascend from the directory being copied.

Here's a summary of how the symlink options are interpreted. The list is in order of precedence, so if your combination of options isn't mentioned, use the first line that is a complete subset of your options:

--copy-links

Turn all symlinks into normal files (leaving no symlinks for any other options to affect).

--links --copy-unsafe-links

Turn all unsafe symlinks into files and duplicate all safe symlinks.

--copy-unsafe-links

Turn all unsafe symlinks into files, noisily skip all safe symlinks.

--links --safe-links

Duplicate safe symlinks and skip unsafe ones.

--links

Duplicate all symlinks.

DIAGNOSTICS

rsync occasionally produces error messages that may seem a little cryptic. The one that seems to cause the most confusion is "protocol version mismatch -- is your shell clean?".

This message is usually caused by your startup scripts or remote shell facility producing unwanted garbage on the stream that rsync is using for its transport. The way to diagnose this problem is to run your remote shell like this:

```
ssh remotehost /bin/true > out.dat
```

then look at out.dat. If everything is working correctly then out.dat should be a zero length file. If you are getting the above error from rsync then you will probably find that out.dat contains some text or data. Look at the contents and try to work out what is producing it. The most common cause is incorrectly configured shell startup scripts (such as .cshrc or .profile) that contain output statements for non-interactive logins.

If you are having trouble debugging filter patterns, then try specifying the **-vv** option. At this level of verbosity rsync will show why each individual file is included or excluded.

EXIT VALUES

0	Success
1	Syntax or usage error
2	Protocol incompatibility
3	Errors selecting input/output files, dirs
4	Requested action not supported: an attempt was made to manipulate 64-bit files on a platform that cannot support them; or an option was specified that is supported by the client and not by the server.
5	Error starting client-server protocol
6	Daemon unable to append to log-file
10	Error in socket I/O
11	Error in file I/O
12	Error in rsync protocol data stream

13	Errors with program diagnostics
14	Error in IPC code
20	Received SIGUSR1 or SIGINT
21	Some error returned by <code>waitpid()</code>
22	Error allocating core memory buffers
23	Partial transfer due to error
24	Partial transfer due to vanished source files
25	The --max-delete limit stopped deletions
30	Timeout in data send/receive
35	Timeout waiting for daemon connection

ENVIRONMENT VARIABLES

CVSIGNORE

The CVSIGNORE environment variable supplements any ignore patterns in .cvsignore files. See the `--cvs-exclude` option for more details.

RSYNC_ICONV

Specify a default `--iconv` setting using this environment variable. (First supported in 3.0.0.)

RSYNC_PROTECT_ARGS

Specify a non-zero numeric value if you want the `--protect-args` option to be enabled by default, or a zero value to make sure that it is disabled by default. (First supported in 3.1.0.)

RSYNC_RSH

The RSYNC_RSH environment variable allows you to override the default shell used as the transport for rsync. Command line options are permitted after the command name, just as in the `-e` option.

RSYNC_PROXY

The RSYNC_PROXY environment variable allows you to redirect your rsync client to use a web proxy when connecting to a rsync daemon. You should set RSYNC_PROXY to a hostname:port pair.

RSYNC_PASSWORD

Setting RSYNC_PASSWORD to the required password allows you to run authenticated rsync connections to an rsync daemon without user intervention. Note that this does not supply a password to a remote shell transport such as ssh; to learn how to do that, consult the remote shell's documentation.

USER or LOGNAME

The USER or LOGNAME environment variables are used to determine the default username sent to an rsync daemon. If neither is set, the username defaults to "nobody".

HOME

The HOME environment variable is used to find the user's default .cvsignore file.

FILES

/etc/rsyncd.conf or rsyncd.conf

SEE ALSO

`rsync-ssl(1)`, `rsyncd.conf(5)`

BUGS

times are transferred as *nix time_t values

When transferring to FAT filesystems rsync may re-sync unmodified files. See the comments on the `--modify-window` option.

file permissions, devices, etc. are transferred as native numerical values

see also the comments on the **--delete** option

Please report bugs! See the web site at <https://rsync.samba.org/>.

VERSION

This man page is current for version 3.2.3 of rsync.

INTERNAL OPTIONS

The options **--server** and **--sender** are used internally by rsync, and should never be typed by a user under normal circumstances. Some awareness of these options may be needed in certain scenarios, such as when setting up a login that can only run an rsync command. For instance, the support directory of the rsync distribution has an example script named `rrsync` (for restricted rsync) that can be used with a restricted ssh login.

CREDITS

rsync is distributed under the GNU General Public License. See the file COPYING for details.

A web site is available at <https://rsync.samba.org/>. The site includes an FAQ-O-Matic which may cover questions unanswered by this manual page.

We would be delighted to hear from you if you like this program. Please contact the mailing-list at rsync@lists.samba.org.

This program uses the excellent zlib compression library written by Jean-loup Gailly and Mark Adler.

THANKS

Special thanks go out to: John Van Essen, Matt McCutchen, Wesley W. Terpstra, David Dykstra, Jos Backus, Sebastian Krahmer, Martin Pool, and our gone-but-not-forgotten compadre, J.W. Schultz.

Thanks also to Richard Brent, Brendan Mackay, Bill Waite, Stephen Rothwell and David Bell. I've probably missed some people, my apologies if I have.

AUTHOR

rsync was originally written by Andrew Tridgell and Paul Mackerras. Many people have later contributed to it. It is currently maintained by Wayne Davison.

Mailing lists for support and development are available at <https://lists.samba.org/>.

NAME

rsyncd.conf – configuration file for rsync in daemon mode

SYNOPSIS

rsyncd.conf

DESCRIPTION

The rsyncd.conf file is the runtime configuration file for rsync when run as an rsync daemon.

The rsyncd.conf file controls authentication, access, logging and available modules.

FILE FORMAT

The file consists of modules and parameters. A module begins with the name of the module in square brackets and continues until the next module begins. Modules contain parameters of the form **name = value**.

The file is line-based -- that is, each newline-terminated line represents either a comment, a module name or a parameter.

Only the first equals sign in a parameter is significant. Whitespace before or after the first equals sign is discarded. Leading, trailing and internal whitespace in module and parameter names is irrelevant. Leading and trailing whitespace in a parameter value is discarded. Internal whitespace within a parameter value is retained verbatim.

Any line **beginning** with a hash (#) is ignored, as are lines containing only whitespace. (If a hash occurs after anything other than leading whitespace, it is considered a part of the line's content.)

Any line ending in a \ is "continued" on the next line in the customary UNIX fashion.

The values following the equals sign in parameters are all either a string (no quotes needed) or a boolean, which may be given as yes/no, 0/1 or true/false. Case is not significant in boolean values, but is preserved in string values.

LAUNCHING THE RSYNC DAEMON

The rsync daemon is launched by specifying the **--daemon** option to rsync.

The daemon must run with root privileges if you wish to use chroot, to bind to a port numbered under 1024 (as is the default 873), or to set file ownership. Otherwise, it must just have permission to read and write the appropriate data, log, and lock files.

You can launch it either via inetd, as a stand-alone daemon, or from an rsync client via a remote shell. If run as a stand-alone daemon then just run the command "**rsync --daemon**" from a suitable startup script.

When run via inetd you should add a line like this to /etc/services:

```
rsync      873/tcp
```

and a single line something like this to /etc/inetd.conf:

```
rsync  stream  tcp  nowait  root  /usr/bin/rsync rsyncd --daemon
```

Replace "/usr/bin/rsync" with the path to where you have rsync installed on your system. You will then need to send inetd a HUP signal to tell it to reread its config file.

Note that you should **not** send the rsync daemon a HUP signal to force it to reread the **rsyncd.conf** file. The file is re-read on each client connection.

GLOBAL PARAMETERS

The first parameters in the file (before a [module] header) are the global parameters. Rsync also allows for the use of a "[global]" module name to indicate the start of one or more global-parameter sections (the name must be lower case).

You may also include any module parameters in the global part of the config file in which case the supplied value will override the default for that parameter.

You may use references to environment variables in the values of parameters. String parameters will have %VAR% references expanded as late as possible (when the string is first used in the program), allowing for

the use of variables that rsync sets at connection time, such as RSYNC_USER_NAME. Non-string parameters (such as true/false settings) are expanded when read from the config file. If a variable does not exist in the environment, or if a sequence of characters is not a valid reference (such as an un-paired percent sign), the raw characters are passed through unchanged. This helps with backward compatibility and safety (e.g. expanding a non-existent %VAR% to an empty string in a path could result in a very unsafe path). The safest way to insert a literal % into a value is to use %%.

motd file

This parameter allows you to specify a "message of the day" to display to clients on each connect. This usually contains site information and any legal notices. The default is no motd file. This can be overridden by the **--dparam=motdfile=FILE** command-line option when starting the daemon.

pid file This parameter tells the rsync daemon to write its process ID to that file. The rsync keeps the file locked so that it can know when it is safe to overwrite an existing file.

The filename can be overridden by the **--dparam=pidfile=FILE** command-line option when starting the daemon.

port You can override the default port the daemon will listen on by specifying this value (defaults to 873). This is ignored if the daemon is being run by inetd, and is superseded by the **--port** command-line option.

address

You can override the default IP address the daemon will listen on by specifying this value. This is ignored if the daemon is being run by inetd, and is superseded by the **--address** command-line option.

socket options

This parameter can provide endless fun for people who like to tune their systems to the utmost degree. You can set all sorts of socket options which may make transfers faster (or slower!). Read the man page for the **setsockopt()** system call for details on some of the options you may be able to set. By default no special socket options are set. These settings can also be specified via the **--sockopts** command-line option.

listen backlog

You can override the default backlog value when the daemon listens for connections. It defaults to 5.

MODULE PARAMETERS

After the global parameters you should define a number of modules, each module exports a directory tree as a symbolic name. Modules are exported by specifying a module name in square brackets [module] followed by the parameters for that module. The module name cannot contain a slash or a closing square bracket. If the name contains whitespace, each internal sequence of whitespace will be changed into a single space, while leading or trailing whitespace will be discarded. Also, the name cannot be "global" as that exact name indicates that global parameters follow (see above).

As with GLOBAL PARAMETERS, you may use references to environment variables in the values of parameters. See the GLOBAL PARAMETERS section for more details.

comment

This parameter specifies a description string that is displayed next to the module name when clients obtain a list of available modules. The default is no comment.

path This parameter specifies the directory in the daemon's filesystem to make available in this module. You must specify this parameter for each module in **rsyncd.conf**.

You may base the path's value off of an environment variable by surrounding the variable name with percent signs. You can even reference a variable that is set by rsync when the user connects. For example, this would use the authorizing user's name in the path:

path = /home/%RSYNC_USER_NAME%

It is fine if the path includes internal spaces — they will be retained verbatim (which means that you shouldn't try to escape them). If your final directory has a trailing space (and this is somehow not something you wish to fix), append a trailing slash to the path to avoid losing the trailing whitespace.

use chroot

If "use chroot" is true, the rsync daemon will chroot to the "path" before starting the file transfer with the client. This has the advantage of extra protection against possible implementation security holes, but it has the disadvantages of requiring super-user privileges, of not being able to follow symbolic links that are either absolute or outside of the new root path, and of complicating the preservation of users and groups by name (see below).

As an additional safety feature, you can specify a dot-dir in the module's "path" to indicate the point where the chroot should occur. This allows rsync to run in a chroot with a non-"/" path for the top of the transfer hierarchy. Doing this guards against unintended library loading (since those absolute paths will not be inside the transfer hierarchy unless you have used an unwise pathname), and lets you setup libraries for the chroot that are outside of the transfer. For example, specifying "/var/rsync/.module1" will chroot to the "/var/rsync" directory and set the inside-chroot path to "/module1". If you had omitted the dot-dir, the chroot would have used the whole path, and the inside-chroot path would have been "/".

When both "use chroot" and "daemon chroot" are false, OR the inside-chroot path of "use chroot" is not "/", rsync will: (1) munge symlinks by default for security reasons (see "munge symlinks" for a way to turn this off, but only if you trust your users), (2) substitute leading slashes in absolute paths with the module's path (so that options such as **--backup-dir**, **--compare-dest**, etc. interpret an absolute path as rooted in the module's "path" dir), and (3) trim ".." path elements from args if rsync believes they would escape the module hierarchy. The default for "use chroot" is true, and is the safer choice (especially if the module is not read-only).

When this parameter is enabled *and* the "name converter" parameter is *not* set, the "numeric ids" parameter will default to being enabled (disabling name lookups). This means that if you manually setup name-lookup libraries in your chroot (instead of using a name converter) that you need to explicitly set **numeric ids = false** for rsync to do name lookups.

If you copy library resources into the module's chroot area, you should protect them through your OS's normal user/group or ACL settings (to prevent the rsync module's user from being able to change them), and then hide them from the user's view via "exclude" (see how in the discussion of that parameter). However, it's easier and safer to setup a name converter.

daemon chroot

This parameter specifies a path to which the daemon will chroot before beginning communication with clients. Module paths (and any "use chroot" settings) will then be related to this one. This lets you choose if you want the whole daemon to be chrooted (with this setting), just the transfers to be chrooted (with "use chroot"), or both. Keep in mind that the "daemon chroot" area may need various OS/lib/etc files installed to allow the daemon to function. By default the daemon runs without any chrooting.

proxy protocol

When this parameter is enabled, all incoming connections must start with a V1 or V2 proxy protocol header. If the header is not found, the connection is closed.

Setting this to **true** requires a proxy server to forward source IP information to rsync, allowing you to log proper IP/host info and make use of client-oriented IP restrictions. The default of **false** means that the IP information comes directly from the socket's metadata. If rsync is not behind a proxy, this should be disabled.

CAUTION: using this option can be dangerous if you do not ensure that only the proxy is allowed to connect to the rsync port. If any non-proxied connections are allowed through, the client will be

able to use a modified rsync to spoof any remote IP address that they desire. You can lock this down using something like iptables **--uid-owner root** rules (for strict localhost access), various firewall rules, or you can require password authorization so that any spoofing by users will not grant extra access.

This setting is global. If you need some modules to require this and not others, then you will need to setup multiple rsync daemon processes on different ports.

name converter

This parameter lets you specify a program that will be run by the rsync daemon to do user & group conversions between names & ids. This script is started prior to any chroot being setup, and runs as the daemon user (not the transfer user). You can specify a fully qualified pathname or a program name that is on the \$PATH.

The program can be used to do normal user & group lookups without having to put any extra files into the chroot area of the module *or* you can do customized conversions.

The nameconvert program has access to all of the environment variables that are described in the section on **pre-xfer exec**. This is useful if you want to customize the conversion using information about the module and/or the copy request.

There is a sample python script in the support dir named "nameconvert" that implements the normal user & group lookups. Feel free to customize it or just use it as documentation to implement your own.

numeric ids

Enabling this parameter disables the mapping of users and groups by name for the current daemon module. This prevents the daemon from trying to load any user/group-related files or libraries. This enabling makes the transfer behave as if the client had passed the **--numeric-ids** command-line option. By default, this parameter is enabled for chroot modules and disabled for non-chroot modules. Also keep in mind that uid/gid preservation requires the module to be running as root (see "uid") or for "fake super" to be configured.

A chroot-enabled module should not have this parameter set to false unless you're using a "name converter" program *or* you've taken steps to ensure that the module has the necessary resources it needs to translate names and that it is not possible for a user to change those resources.

munge symlinks

This parameter tells rsync to modify all symlinks in the same way as the (non-daemon-affecting) **--munge-links** command-line option (using a method described below). This should help protect your files from user trickery when your daemon module is writable. The default is disabled when "use chroot" is on with an inside-chroot path of "/", OR if "daemon chroot" is on, otherwise it is enabled.

If you disable this parameter on a daemon that is not read-only, there are tricks that a user can play with uploaded symlinks to access daemon-excluded items (if your module has any), and, if "use chroot" is off, rsync can even be tricked into showing or changing data that is outside the module's path (as access-permissions allow).

The way rsync disables the use of symlinks is to prefix each one with the string "/rsyncd-munged/". This prevents the links from being used as long as that directory does not exist. When this parameter is enabled, rsync will refuse to run if that path is a directory or a symlink to a directory. When using the "munge symlinks" parameter in a chroot area that has an inside-chroot path of "/", you should add "/rsyncd-munged/" to the exclude setting for the module so that a user can't try to create it.

Note: rsync makes no attempt to verify that any pre-existing symlinks in the module's hierarchy are as safe as you want them to be (unless, of course, it just copied in the whole hierarchy). If you setup an rsync daemon on a new area or locally add symlinks, you can manually protect your symlinks from being abused by prefixing "/rsyncd-munged/" to the start of every symlink's value. There is a perl script in the support directory of the source code named "munge-symlinks" that can

be used to add or remove this prefix from your symlinks.

When this parameter is disabled on a writable module and "use chroot" is off (or the inside-chroot path is not "/"), incoming symlinks will be modified to drop a leading slash and to remove ".." path elements that rsync believes will allow a symlink to escape the module's hierarchy. There are tricky ways to work around this, though, so you had better trust your users if you choose this combination of parameters.

charset

This specifies the name of the character set in which the module's filenames are stored. If the client uses an **--iconv** option, the daemon will use the value of the "charset" parameter regardless of the character set the client actually passed. This allows the daemon to support charset conversion in a chroot module without extra files in the chroot area, and also ensures that name-transliteration is done in a consistent manner. If the "charset" parameter is not set, the **--iconv** option is refused, just as if "iconv" had been specified via "refuse options".

If you wish to force users to always use **--iconv** for a particular module, add "no-iconv" to the "refuse options" parameter. Keep in mind that this will restrict access to your module to very new rsync clients.

max connections

This parameter allows you to specify the maximum number of simultaneous connections you will allow. Any clients connecting when the maximum has been reached will receive a message telling them to try later. The default is 0, which means no limit. A negative value disables the module. See also the "lock file" parameter.

log file When the "log file" parameter is set to a non-empty string, the rsync daemon will log messages to the indicated file rather than using syslog. This is particularly useful on systems (such as AIX) where **syslog()** doesn't work for chrooted programs. The file is opened before **chroot()** is called, allowing it to be placed outside the transfer. If this value is set on a per-module basis instead of globally, the global log will still contain any authorization failures or config-file error messages.

If the daemon fails to open the specified file, it will fall back to using syslog and output an error about the failure. (Note that the failure to open the specified log file used to be a fatal error.)

This setting can be overridden by using the **--log-file=FILE** or **--dparam=logfile=FILE** command-line options. The former overrides all the log-file parameters of the daemon and all module settings. The latter sets the daemon's log file and the default for all the modules, which still allows modules to override the default setting.

syslog facility

This parameter allows you to specify the syslog facility name to use when logging messages from the rsync daemon. You may use any standard syslog facility name which is defined on your system. Common names are auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0, local1, local2, local3, local4, local5, local6 and local7. The default is daemon. This setting has no effect if the "log file" setting is a non-empty string (either set in the per-modules settings, or inherited from the global settings).

syslog tag

This parameter allows you to specify the syslog tag to use when logging messages from the rsync daemon. The default is "rsyncd". This setting has no effect if the "log file" setting is a non-empty string (either set in the per-modules settings, or inherited from the global settings).

For example, if you wanted each authenticated user's name to be included in the syslog tag, you could do something like this:

```
syslog tag = rsyncd.%RSYNC_USER_NAME%
```

max verbosity

This parameter allows you to control the maximum amount of verbose information that you'll allow the daemon to generate (since the information goes into the log file). The default is 1, which allows the client to request one level of verbosity.

This also affects the user's ability to request higher levels of **--info** and **--debug** logging. If the max value is 2, then no info and/or debug value that is higher than what would be set by **-vv** will be honored by the daemon in its logging. To see how high of a verbosity level you need to accept for a particular info/debug level, refer to **rsync --info=help** and **rsync --debug=help**. For instance, it takes max-verbosity 4 to be able to output debug TIME2 and FLIST3.

lock file

This parameter specifies the file to use to support the "max connections" parameter. The rsync daemon uses record locking on this file to ensure that the max connections limit is not exceeded for the modules sharing the lock file. The default is **/var/run/rsyncd.lock**.

read only

This parameter determines whether clients will be able to upload files or not. If "read only" is true then any attempted uploads will fail. If "read only" is false then uploads will be possible if file permissions on the daemon side allow them. The default is for all modules to be read only.

Note that "auth users" can override this setting on a per-user basis.

write only

This parameter determines whether clients will be able to download files or not. If "write only" is true then any attempted downloads will fail. If "write only" is false then downloads will be possible if file permissions on the daemon side allow them. The default is for this parameter to be disabled.

Helpful hint: you probably want to specify "refuse options = delete" for a write-only module.

open noatime

When set to True, this parameter tells the rsync daemon to open files with the O_NOATIME flag (on systems that support it) to avoid changing the access time of the files that are being transferred. If your OS does not support the O_NOATIME flag then rsync will silently ignore this option. Note also that some filesystems are mounted to avoid updating the atime on read access even without the O_NOATIME flag being set.

When set to False, this parameters ensures that files on the server are not opened with O_NOATIME.

When set to Unset (the default) the user controls the setting via **--open-noatime**.

list This parameter determines whether this module is listed when the client asks for a listing of available modules. In addition, if this is false, the daemon will pretend the module does not exist when a client denied by "hosts allow" or "hosts deny" attempts to access it. Realize that if "reverse lookup" is disabled globally but enabled for the module, the resulting reverse lookup to a potentially client-controlled DNS server may still reveal to the client that it hit an existing module. The default is for modules to be listable.

uid This parameter specifies the user name or user ID that file transfers to and from that module should take place as when the daemon was run as root. In combination with the "gid" parameter this determines what file permissions are available. The default when run by a super-user is to switch to the system's "nobody" user. The default for a non-super-user is to not try to change the user. See also the "gid" parameter.

The RSYNC_USER_NAME environment variable may be used to request that rsync run as the authorizing user. For example, if you want a rsync to run as the same user that was received for the rsync authentication, this setup is useful:

```
uid = %RSYNC_USER_NAME%
gid = *
```

gid This parameter specifies one or more group names/IDs that will be used when accessing the module. The first one will be the default group, and any extra ones be set as supplemental groups. You may also specify a "*" as the first gid in the list, which will be replaced by all the normal groups for the transfer's user (see "uid"). The default when run by a super-user is to switch to your OS's

"nobody" (or perhaps "nogroup") group with no other supplementary groups. The default for a non-super-user is to not change any group attributes (and indeed, your OS may not allow a non-super-user to try to change their group settings).

The specified list is normally split into tokens based on spaces and commas. However, if the list starts with a comma, then the list is only split on commas, which allows a group name to contain a space. In either case any leading and/or trailing whitespace is removed from the tokens and empty tokens are ignored.

daemon uid

This parameter specifies a uid under which the daemon will run. The daemon usually runs as user root, and when this is left unset the user is left unchanged. See also the "uid" parameter.

daemon gid

This parameter specifies a gid under which the daemon will run. The daemon usually runs as group root, and when this is left unset, the group is left unchanged. See also the "gid" parameter.

fake super

Setting "fake super = yes" for a module causes the daemon side to behave as if the **--fake-super** command-line option had been specified. This allows the full attributes of a file to be stored without having to have the daemon actually running as root.

filter

The daemon has its own filter chain that determines what files it will let the client access. This chain is not sent to the client and is independent of any filters the client may have specified. Files excluded by the daemon filter chain (**daemon-excluded** files) are treated as non-existent if the client tries to pull them, are skipped with an error message if the client tries to push them (triggering exit code 23), and are never deleted from the module. You can use daemon filters to prevent clients from downloading or tampering with private administrative files, such as files you may add to support uid/gid name translations.

The daemon filter chain is built from the "filter", "include from", "include", "exclude from", and "exclude" parameters, in that order of priority. Anchored patterns are anchored at the root of the module. To prevent access to an entire subtree, for example, "/secret", you **must** exclude everything in the subtree; the easiest way to do this is with a triple-star pattern like "/secret/**".

The "filter" parameter takes a space-separated list of daemon filter rules, though it is smart enough to know not to split a token at an internal space in a rule (e.g. "- /foo - /bar" is parsed as two rules). You may specify one or more merge-file rules using the normal syntax. Only one "filter" parameter can apply to a given module in the config file, so put all the rules you want in a single parameter. Note that per-directory merge-file rules do not provide as much protection as global rules, but they can be used to make **--delete** work better during a client download operation if the per-dir merge files are included in the transfer and the client requests that they be used.

exclude

This parameter takes a space-separated list of daemon exclude patterns. As with the client **--exclude** option, patterns can be qualified with "-" or "+" to explicitly indicate exclude/include. Only one "exclude" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

include

Use an "include" to override the effects of the "exclude" parameter. Only one "include" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

exclude from

This parameter specifies the name of a file on the daemon that contains daemon exclude patterns, one per line. Only one "exclude from" parameter can apply to a given module; if you have multiple exclude-from files, you can specify them as a merge file in the "filter" parameter. See the "filter" parameter for a description of how excluded files affect the daemon.

include from

Analogue of "exclude from" for a file of daemon include patterns. Only one "include from" parameter can apply to a given module. See the "filter" parameter for a description of how excluded files affect the daemon.

incoming chmod

This parameter allows you to specify a set of comma-separated chmod strings that will affect the permissions of all incoming files (files that are being received by the daemon). These changes happen after all other permission calculations, and this will even override destination-default and/or existing permissions when the client does not specify **--perms**. See the description of the **--chmod** rsync option and the **chmod(1)** manpage for information on the format of this string.

outgoing chmod

This parameter allows you to specify a set of comma-separated chmod strings that will affect the permissions of all outgoing files (files that are being sent out from the daemon). These changes happen first, making the sent permissions appear to be different than those stored in the filesystem itself. For instance, you could disable group write permissions on the server while having it appear to be on to the clients. See the description of the **--chmod** rsync option and the **chmod(1)** manpage for information on the format of this string.

auth users

This parameter specifies a comma and/or space-separated list of authorization rules. In its simplest form, you list the usernames that will be allowed to connect to this module. The usernames do not need to exist on the local system. The rules may contain shell wildcard characters that will be matched against the username provided by the client for authentication. If "auth users" is set then the client will be challenged to supply a username and password to connect to the module. A challenge response authentication protocol is used for this exchange. The plain text usernames and passwords are stored in the file specified by the "secrets file" parameter. The default is for all users to be able to connect without a password (this is called "anonymous rsync").

In addition to username matching, you can specify groupname matching via a '@' prefix. When using groupname matching, the authenticating username must be a real user on the system, or it will be assumed to be a member of no groups. For example, specifying "@rsync" will match the authenticating user if the named user is a member of the rsync group.

Finally, options may be specified after a colon (:). The options allow you to "deny" a user or a group, set the access to "ro" (read-only), or set the access to "rw" (read/write). Setting an auth-rule-specific ro/rw setting overrides the module's "read only" setting.

Be sure to put the rules in the order you want them to be matched, because the checking stops at the first matching user or group, and that is the only auth that is checked. For example:

```
auth users = joe:deny @guest:deny admin:rw @rsync:ro susan joe sam
```

In the above rule, user joe will be denied access no matter what. Any user that is in the group "guest" is also denied access. The user "admin" gets access in read/write mode, but only if the admin user is not in group "guest" (because the admin user-matching rule would never be reached if the user is in group "guest"). Any other user who is in group "rsync" will get read-only access. Finally, users susan, joe, and sam get the ro/rw setting of the module, but only if the user didn't match an earlier group-matching rule.

If you need to specify a user or group name with a space in it, start your list with a comma to indicate that the list should only be split on commas (though leading and trailing whitespace will also be removed, and empty entries are just ignored). For example:

```
auth users = , joe:deny, @Some Group:deny, admin:rw, @RO Group:ro
```

See the description of the secrets file for how you can have per-user passwords as well as per-group passwords. It also explains how a user can authenticate using their user password or (when applicable) a group password, depending on what rule is being authenticated.

See also the section entitled "USING RSYNC-DAEMON FEATURES VIA A REMOTE SHELL CONNECTION" in **rsync(1)** for information on how handle an rsyncd.conf-level username that differs from the remote-shell-level username when using a remote shell to connect to an rsync daemon.

secrets file

This parameter specifies the name of a file that contains the username:password and/or @group-name:password pairs used for authenticating this module. This file is only consulted if the "auth users" parameter is specified. The file is line-based and contains one name:password pair per line. Any line has a hash (#) as the very first character on the line is considered a comment and is skipped. The passwords can contain any characters but be warned that many operating systems limit the length of passwords that can be typed at the client end, so you may find that passwords longer than 8 characters don't work.

The use of group-specific lines are only relevant when the module is being authorized using a matching "@groupname" rule. When that happens, the user can be authorized via either their "username:password" line or the "@groupname:password" line for the group that triggered the authentication.

It is up to you what kind of password entries you want to include, either users, groups, or both. The use of group rules in "auth users" does not require that you specify a group password if you do not want to use shared passwords.

There is no default for the "secrets file" parameter, you must choose a name (such as /etc/rsyncd.secrets). The file must normally not be readable by "other"; see "strict modes". If the file is not found or is rejected, no logins for a "user auth" module will be possible.

strict modes

This parameter determines whether or not the permissions on the secrets file will be checked. If "strict modes" is true, then the secrets file must not be readable by any user ID other than the one that the rsync daemon is running under. If "strict modes" is false, the check is not performed. The default is true. This parameter was added to accommodate rsync running on the Windows operating system.

hosts allow

This parameter allows you to specify a list of comma- and/or whitespace-separated patterns that are matched against a connecting client's hostname and IP address. If none of the patterns match, then the connection is rejected.

Each pattern can be in one of six forms:

- o a dotted decimal IPv4 address of the form a.b.c.d, or an IPv6 address of the form a:b:c::d:e:f. In this case the incoming machine's IP address must match exactly.
- o an address/mask in the form ipaddr/n where ipaddr is the IP address and n is the number of one bits in the netmask. All IP addresses which match the masked IP address will be allowed in.
- o an address/mask in the form ipaddr/maskaddr where ipaddr is the IP address and maskaddr is the netmask in dotted decimal notation for IPv4, or similar for IPv6, e.g. ffff:ffff:ffff:ffff:: instead of /64. All IP addresses which match the masked IP address will be allowed in.
- o a hostname pattern using wildcards. If the hostname of the connecting IP (as determined by a reverse lookup) matches the wildcarded name (using the same rules as normal unix filename matching), the client is allowed in. This only works if "reverse lookup" is enabled (the default).
- o a hostname. A plain hostname is matched against the reverse DNS of the connecting IP (if "reverse lookup" is enabled), and/or the IP of the given hostname is matched against the connecting IP (if "forward lookup" is enabled, as it is by default). Any match will be allowed in.

- o an '@' followed by a netgroup name, which will match if the reverse DNS of the connecting IP is in the specified netgroup.

Note IPv6 link-local addresses can have a scope in the address specification:

```
fe80::1%link1
fe80::%link1/64
fe80::%link1/ffff:ffff:ffff::
```

You can also combine "hosts allow" with "hosts deny" as a way to add exceptions to your deny list. When both parameters are specified, the "hosts allow" parameter is checked first and a match results in the client being able to connect. A non-allowed host is then matched against the "hosts deny" list to see if it should be rejected. A host that does not match either list is allowed to connect.

The default is no "hosts allow" parameter, which means all hosts can connect.

hosts deny

This parameter allows you to specify a list of comma- and/or whitespace-separated patterns that are matched against a connecting clients hostname and IP address. If the pattern matches then the connection is rejected. See the "hosts allow" parameter for more information.

The default is no "hosts deny" parameter, which means all hosts can connect.

reverse lookup

Controls whether the daemon performs a reverse lookup on the client's IP address to determine its hostname, which is used for "hosts allow" & "hosts deny" checks and the "%h" log escape. This is enabled by default, but you may wish to disable it to save time if you know the lookup will not return a useful result, in which case the daemon will use the name "UNDETERMINED" instead.

If this parameter is enabled globally (even by default), rsync performs the lookup as soon as a client connects, so disabling it for a module will not avoid the lookup. Thus, you probably want to disable it globally and then enable it for modules that need the information.

forward lookup

Controls whether the daemon performs a forward lookup on any hostname specified in an hosts allow/deny setting. By default this is enabled, allowing the use of an explicit hostname that would not be returned by reverse DNS of the connecting IP.

ignore errors

This parameter tells rsyncd to ignore I/O errors on the daemon when deciding whether to run the delete phase of the transfer. Normally rsync skips the --**delete** step if any I/O errors have occurred in order to prevent disastrous deletion due to a temporary resource shortage or other I/O error. In some cases this test is counter productive so you can use this parameter to turn off this behavior.

ignore nonreadable

This tells the rsync daemon to completely ignore files that are not readable by the user. This is useful for public archives that may have some non-readable files among the directories, and the sysadmin doesn't want those files to be seen at all.

transfer logging

This parameter enables per-file logging of downloads and uploads in a format somewhat similar to that used by ftp daemons. The daemon always logs the transfer at the end, so if a transfer is aborted, no mention will be made in the log file.

If you want to customize the log lines, see the "log format" parameter.

log format

This parameter allows you to specify the format used for logging file transfers when transfer logging is enabled. The format is a text string containing embedded single-character escape sequences prefixed with a percent (%) character. An optional numeric field width may also be specified between the percent and the escape letter (e.g. "%-50n %8I %07p"). In addition, one or more apostrophes may be specified prior to a numerical escape to indicate that the numerical value

should be made more human-readable. The 3 supported levels are the same as for the **--human-readable** command-line option, though the default is for human-readability to be off. Each added apostrophe increases the level (e.g. "%'"l %'b %'f").

The default log format is "%o %h [%a] %m (%u) %f %l", and a "%t [%p]" is always prefixed when using the "log file" parameter. (A perl script that will summarize this default log format is included in the rsync source code distribution in the "support" subdirectory: rsyncstats.)

The single-character escapes that are understood are as follows:

- o %a the remote IP address (only available for a daemon)
- o %b the number of bytes actually transferred
- o %B the permission bits of the file (e.g. rwxrwxrwt)
- o %c the total size of the block checksums received for the basis file (only when sending)
- o %C the full-file checksum if it is known for the file. For older rsync protocols/versions, the checksum was salted, and is thus not a useful value (and is not displayed when that is the case). For the checksum to output for a file, either the **--checksum** option must be in-effect or the file must have been transferred without a salted checksum being used. See the **--checksum-choice** option for a way to choose the algorithm.
- o %f the filename (long form on sender; no trailing "/")
- o %G the gid of the file (decimal) or "DEFAULT"
- o %h the remote host name (only available for a daemon)
- o %i an itemized list of what is being updated
- o %l the length of the file in bytes
- o %L the string "-> SYMLINK", "=> HARDLINK", or "" (where **SYMLINK** or **HARDLINK** is a filename)
- o %m the module name
- o %M the last-modified time of the file
- o %n the filename (short form; trailing "/" on dir)
- o %o the operation, which is "send", "recv", or "del." (the latter includes the trailing period)
- o %p the process ID of this rsync session
- o %P the module path
- o %t the current date time
- o %u the authenticated username or an empty string
- o %U the uid of the file (decimal)

For a list of what the characters mean that are output by "%i", see the **--itemize-changes** option in the rsync manpage.

Note that some of the logged output changes when talking with older rsync versions. For instance, deleted files were only output as verbose messages prior to rsync 2.6.4.

timeout

This parameter allows you to override the clients choice for I/O timeout for this module. Using this parameter you can ensure that rsync won't wait on a dead client forever. The timeout is specified in seconds. A value of zero means no timeout and is the default. A good choice for anonymous rsync daemons may be 600 (giving a 10 minute timeout).

refuse options

This parameter allows you to specify a space-separated list of rsync command-line options that will be refused by your rsync daemon. You may specify the full option name, its one-letter

abbreviation, or a wild-card string that matches multiple options. Beginning in 3.2.0, you can also negate a match term by starting it with a "!".

When an option is refused, the daemon prints an error message and exits.

For example, this would refuse **--checksum** (**-c**) and all the various delete options:

```
refuse options = c delete
```

The reason the above refuses all delete options is that the options imply **--delete**, and implied options are refused just like explicit options.

The use of a negated match allows you to fine-tune your refusals after a wild-card, such as this:

```
refuse options = delete-* !delete-during
```

Negated matching can also turn your list of refused options into a list of accepted options. To do this, begin the list with a "*" (to refuse all options) and then specify one or more negated matches to accept. For example:

```
refuse options = * !a !v !compress*
```

Don't worry that the "*" will refuse certain vital options such as **--dry-run**, **--server**, **--no-iconv**, **--protect-args**, etc. These important options are not matched by wild-card, so they must be overridden by their exact name. For instance, if you're forcing iconv transfers you could use something like this:

```
refuse options = * no-iconv !a !v
```

As an additional aid (beginning in 3.2.0), refusing (or "**!refusing**") the "a" or "archive" option also affects all the options that the **--archive** option implies (**-rdlptgoD**), but only if the option is matched explicitly (not using a wildcard). If you want to do something tricky, you can use "**archive***" to avoid this side-effect, but keep in mind that no normal rsync client ever sends the actual archive option to the server.

As an additional safety feature, the refusal of "delete" also refuses **remove-source-files** when the daemon is the sender; if you want the latter without the former, instead refuse "**delete-***" as that refuses all the delete modes without affecting **remove-source-files**. (Keep in mind that the client's **--delete** option typically results in **--delete-during**.)

When un-refusing delete options, you should either specify "**!delete***" (to accept all delete options) or specify a limited set that includes "delete", such as:

```
refuse options = * !a !delete !delete-during
```

... whereas this accepts any delete option except **--delete-after**:

```
refuse options = * !a !delete* delete-after
```

A note on refusing "compress" -- it is better to set the "dont compress" daemon parameter to "*" because that disables compression silently instead of returning an error that forces the client to remove the **-z** option.

If you are un-refusing the compress option, you probably want to match "**!compress***" so that you also accept the **--compress-level** option.

Note that the "copy-devices" & "write-devices" options are refused by default, but they can be explicitly accepted with "**!copy-devices**" and/or "**!write-devices**". The options "log-file" and "log-file-format" are forcibly refused and cannot be accepted.

Here are all the options that are not matched by wild-cards:

- o **--server**: Required for rsync to even work.
- o **--rsh**, **-e**: Required to convey compatibility flags to the server.
- o **--out-format**: This is required to convey output behavior to a remote receiver. While rsync passes the older alias **--log-format** for compatibility reasons, this options should

not be confused with **--log-file-format**.

- o **--sender**: Use "write only" parameter instead of refusing this.
- o **--dry-run, -n**: Who would want to disable this?
- o **--protect-args, -s**: This actually makes transfers safer.
- o **--from0, -0**: Makes it easier to accept/refuse **--files-from** without affecting this helpful modifier.
- o **--iconv**: This is auto-disabled based on "charset" parameter.
- o **--no-iconv**: Most transfers use this option.
- o **--checksum-seed**: Is a fairly rare, safe option.
- o **--write-devices**: Is non-wild but also auto-disabled.

dont compress

This parameter allows you to select filenames based on wildcard patterns that should not be compressed when pulling files from the daemon (no analogous parameter exists to govern the pushing of files to a daemon). Compression can be expensive in terms of CPU usage, so it is usually good to not try to compress files that won't compress well, such as already compressed files.

The "dont compress" parameter takes a space-separated list of case-insensitive wildcard patterns. Any source filename matching one of the patterns will be compressed as little as possible during the transfer. If the compression algorithm has an "off" level (such as zlib/zlibx) then no compression occurs for those files. Other algorithms have the level minimized to reduce the CPU usage as much as possible.

See the **--skip-compress** parameter in the **rsync(1)** manpage for the list of file suffixes that are not compressed by default. Specifying a value for the "dont compress" parameter changes the default when the daemon is the sender.

early exec, pre-xfer exec, post-xfer exec

You may specify a command to be run in the early stages of the connection, or right before and/or after the transfer. If the **early exec** or **pre-xfer exec** command returns an error code, the transfer is aborted before it begins. Any output from the **pre-xfer exec** command on stdout (up to several KB) will be displayed to the user when aborting, but is *not* displayed if the script returns success. The other programs cannot send any text to the user. All output except for the **pre-xfer exec** stdout goes to the corresponding daemon's stdout/stderr, which is typically discarded. See the **--no-detach** option for a way to see the daemon's output, which can assist with debugging.

Note that the **early exec** command runs before any part of the transfer request is known except for the module name. This helper script can be used to setup a disk mount or decrypt some data into a module dir, but you may need to use **lock file** and **max connections** to avoid concurrency issues. If the client rsync specified the **--early-input=FILE** option, it can send up to about 5K of data to the stdin of the early script. The stdin will otherwise be empty.

Note that the **post-xfer exec** command is still run even if one of the other scripts returns an error code. The **pre-xfer exec** command will *not* be run, however, if the **early exec** command fails.

The following environment variables will be set, though some are specific to the pre-xfer or the post-xfer environment:

- o **RSYNC_MODULE_NAME**: The name of the module being accessed.
- o **RSYNC_MODULE_PATH**: The path configured for the module.
- o **RSYNC_HOST_ADDR**: The accessing host's IP address.
- o **RSYNC_HOST_NAME**: The accessing host's name.
- o **RSYNC_USER_NAME**: The accessing user's name (empty if no user).

- o **RSYNC_PID**: A unique number for this transfer.
- o **RSYNC_REQUEST**: (pre-xfer only) The module/path info specified by the user. Note that the user can specify multiple source files, so the request can be something like "mod/path1 mod/path2", etc.
- o **RSYNC_ARG#**: (pre-xfer only) The pre-request arguments are set in these numbered values. RSYNC_ARG0 is always "rsyncd", followed by the options that were used in RSYNC_ARG1, and so on. There will be a value of "." indicating that the options are done and the path args are beginning -- these contain similar information to RSYNC_REQUEST, but with values separated and the module name stripped off.
- o **RSYNC_EXIT_STATUS**: (post-xfer only) the server side's exit value. This will be 0 for a successful run, a positive value for an error that the server generated, or a -1 if rsync failed to exit properly. Note that an error that occurs on the client side does not currently get sent to the server side, so this is not the final exit status for the whole transfer.
- o **RSYNC_RAW_STATUS**: (post-xfer only) the raw exit value from `waitpid()`.

Even though the commands can be associated with a particular module, they are run using the permissions of the user that started the daemon (not the module's uid/gid setting) without any chroot restrictions.

These settings honor 2 environment variables: use RSYNC_SHELL to set a shell to use when running the command (which otherwise uses your `system()` call's default shell), and use RSYNC_NO_XFER_EXEC to disable both options completely.

CONFIG DIRECTIVES

There are currently two config directives available that allow a config file to incorporate the contents of other files: **&include** and **&merge**. Both allow a reference to either a file or a directory. They differ in how segregated the file's contents are considered to be.

The **&include** directive treats each file as more distinct, with each one inheriting the defaults of the parent file, starting the parameter parsing as globals/defaults, and leaving the defaults unchanged for the parsing of the rest of the parent file.

The **&merge** directive, on the other hand, treats the file's contents as if it were simply inserted in place of the directive, and thus it can set parameters in a module started in another file, can affect the defaults for other files, etc.

When an **&include** or **&merge** directive refers to a directory, it will read in all the `*.conf` or `*.inc` files (respectively) that are contained inside that directory (without any recursive scanning), with the files sorted into alpha order. So, if you have a directory named "rsyncd.d" with the files "foo.conf", "bar.conf", and "baz.conf" inside it, this directive:

```
&include /path/rsyncd.d
```

would be the same as this set of directives:

```
&include /path/rsyncd.d/bar.conf
&include /path/rsyncd.d/baz.conf
&include /path/rsyncd.d/foo.conf
```

except that it adjusts as files are added and removed from the directory.

The advantage of the **&include** directive is that you can define one or more modules in a separate file without worrying about unintended side-effects between the self-contained module files.

The advantage of the **&merge** directive is that you can load config snippets that can be included into multiple module definitions, and you can also set global values that will affect connections (such as **motd file**), or globals that will affect other include files.

For example, this is a useful `/etc/rsyncd.conf` file:

```
port = 873
```

```
log file = /var/log/rsync.log
pid file = /var/lock/rsync.lock
```

```
&merge /etc/rsyncd.d
&include /etc/rsyncd.d
```

This would merge any **/etc/rsyncd.d/*.inc** files (for global values that should stay in effect), and then include any **/etc/rsyncd.d/*.conf** files (defining modules without any global-value cross-talk).

AUTHENTICATION STRENGTH

The authentication protocol used in rsync is a 128 bit MD4 based challenge response system. This is fairly weak protection, though (with at least one brute-force hash-finding algorithm publicly available), so if you want really top-quality security, then I recommend that you run rsync over ssh. (Yes, a future version of rsync will switch over to a stronger hashing method.)

Also note that the rsync daemon protocol does not currently provide any encryption of the data that is transferred over the connection. Only authentication is provided. Use ssh as the transport if you want encryption.

You can also make use of SSL/TLS encryption if you put rsync behind an SSL proxy.

SSL/TLS Daemon Setup

When setting up an rsync daemon for access via SSL/TLS, you will need to configure a proxy (such as haproxy or nginx) as the front-end that handles the encryption.

- o You should limit the access to the backend-rsyncd port to only allow the proxy to connect. If it is on the same host as the proxy, then configuring it to only listen on localhost is a good idea.
- o You should consider turning on the **proxy protocol** parameter if your proxy supports sending that information. The examples below assume that this is enabled.

An example haproxy setup is as follows:

```
frontend fe_rsync-ssl
    bind :::874 ssl crt /etc/letsencrypt/example.com/combined.pem
    mode tcp
    use_backend be_rsync

backend be_rsync
    mode tcp
    server local-rsync 127.0.0.1:873 check send-proxy
```

An example nginx proxy setup is as follows:

```
stream {
    server {
        listen 874 ssl;
        listen [::]:874 ssl;

        ssl_certificate /etc/letsencrypt/example.com/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/example.com/privkey.pem;

        proxy_pass localhost:873;
        proxy_protocol on; # Requires "proxy protocol = true"
        proxy_timeout 1m;
        proxy_connect_timeout 5s;
    }
}
```

EXAMPLES

A simple rsyncd.conf file that allow anonymous rsync to a ftp area at **/home/ftp** would be:

```
[ftp]
```

```
path = /home/ftp
comment = ftp export area
```

A more sophisticated example would be:

```
uid = nobody
gid = nobody
use chroot = yes
max connections = 4
syslog facility = local5
pid file = /var/run/rsyncd.pid
```

```
[ftp]
path = /var/ftp./pub
comment = whole ftp area (approx 6.1 GB)
```

```
[sambaftp]
path = /var/ftp./pub/samba
comment = Samba ftp area (approx 300 MB)
```

```
[rsyncftp]
path = /var/ftp./pub/rsync
comment = rsync ftp area (approx 6 MB)
```

```
[sambawww]
path = /public_html/samba
comment = Samba WWW pages (approx 240 MB)
```

```
[cvs]
path = /data/cvs
comment = CVS repository (requires authentication)
auth users = tridge, susan
secrets file = /etc/rsyncd.secrets
```

The /etc/rsyncd.secrets file would look something like this:

```
tridge:mypass
susan:herpass
```

FILES

/etc/rsyncd.conf or rsyncd.conf

SEE ALSO

rsync(1), rsync-ssl(1)

BUGS

Please report bugs! The rsync bug tracking system is online at <https://rsync.samba.org/>.

VERSION

This man page is current for version 3.2.3 of rsync.

CREDITS

rsync is distributed under the GNU General Public License. See the file COPYING for details.

The primary ftp site for rsync is <ftp://rsync.samba.org/pub/rsync>

A web site is available at <https://rsync.samba.org/>.

We would be delighted to hear from you if you like this program.

This program uses the zlib compression library written by Jean-loup Gailly and Mark Adler.

THANKS

Thanks to Warren Stanley for his original idea and patch for the rsync daemon. Thanks to Karsten Thygesen for his many suggestions and documentation!

AUTHOR

rsync was written by Andrew Tridgell and Paul Mackerras. Many people have later contributed to it.

Mailing lists for support and development are available at <https://lists.samba.org/>.

samba(7) - Linux man page

Name

samba - A Windows SMB/CIFS fileserver for UNIX

Synopsis

samba

Description

The Samba software suite is a collection of programs that implements the Server Message Block (commonly abbreviated as SMB) protocol for UNIX systems. This protocol is sometimes also referred to as the Common Internet File System (CIFS). For a more thorough description, see <http://www.ubiqx.org/cifs/>. Samba also implements the NetBIOS protocol in nmbd.

smbd(8)

The smbd daemon provides the file and print services to SMB clients, such as Windows 95/98, Windows NT, Windows for Workgroups or LanManager. The configuration file for this daemon is described in **smb.conf(5)**

nmbd(8)

The nmbd daemon provides NetBIOS nameservice and browsing support. The configuration file for this daemon is described in **smb.conf(5)**

smbclient(1)

The smbclient program implements a simple ftp-like client. This is useful for accessing SMB shares on other compatible servers (such as Windows NT), and can also be used to allow a UNIX box to print to a printer attached to any SMB server (such as a PC running Windows NT).

testparm(1)

The testparm utility is a simple syntax checker for Samba's **smb.conf(5)** configuration file.

smbstatus(1)

The smbstatus tool provides access to information about the current connections to smbd.

nmblookup(1)

The nmblookup tools allows NetBIOS name queries to be made from a UNIX host.

smbpasswd(8)

The smbpasswd command is a tool for changing LanMan and Windows NT password hashes on Samba and Windows NT servers.

smbcacls(1)

The smbcacls command is a tool to set ACL's on remote CIFS servers.

smbtree(1)

The smbtree command is a text-based network neighborhood tool.

smbtar(1)

The smbtar can make backups of data on CIFS/SMB servers.

smbspool(8)

smbspool is a helper utility for printing on printers connected to CIFS servers.

smbcontrol(1)

smbcontrol is a utility that can change the behaviour of running samba daemons.

rpcclient(1)

rpcclient is a utility that can be used to execute RPC commands on remote CIFS servers.

pdbedit(8)

The pdbedit command can be used to maintain the local user database on a samba server.

findsmb(1)

The findsmb command can be used to find SMB servers on the local network.

net(8)

The net command is supposed to work similar to the DOS/Windows NET.EXE command.

swat(8)

swat is a web-based interface to configuring smb.conf.

winbindd(8)

winbindd is a daemon that is used for integrating authentication and the user database into unix.

wbinfo(1)

wbinfo is a utility that retrieves and stores information related to winbind.

profiles(1)

profiles is a command-line utility that can be used to replace all occurrences of a certain SID with another SID.

log2pcap(1)

log2pcap is a utility for generating pcap trace files from Samba log files.

vfstest(1)

vfstest is a utility that can be used to test vfs modules.

ntlm_auth(1)

ntlm_auth is a helper-utility for external programs wanting to do NTLM-authentication.

smbcquotas(1)

smbcquotas is a tool that can set remote QUOTA's on server with NTFS 5.

Components

The Samba suite is made up of several components. Each component is described in a separate manual page. It is strongly recommended that you read the documentation that comes with Samba and the manual pages of those components that you use. If the manual

pages and documents aren't clear enough then please visit <http://devel.samba.org> for information on how to file a bug report or submit a patch.

If you require help, visit the Samba webpage at <http://www.samba.org/> and explore the many option available to you.

Availability

The Samba software suite is licensed under the GNU Public License(GPL). A copy of that license should have come with the package in the file COPYING. You are encouraged to distribute copies of the Samba suite, but please obey the terms of this license.

The latest version of the Samba suite can be obtained via anonymous ftp from samba.org in the directory pub/samba/. It is also available on several mirror sites worldwide.

You may also find useful information about Samba on the newsgroup comp.protocol.smb and the Samba mailing list. Details on how to join the mailing list are given in the README file that comes with Samba.

If you have access to a WWW viewer (such as Mozilla or Konqueror) then you will also find lots of useful information, including back issues of the Samba mailing list, at <http://lists.samba.org>.

Version

This man page is correct for version 3 of the Samba suite.

Contributions

If you wish to contribute to the Samba project, then I suggest you join the Samba mailing list at <http://lists.samba.org>.

If you have patches to submit, visit <http://devel.samba.org/> for information on how to do it properly. We prefer patches in diff -u format.

Contributors

Contributors to the project are now too numerous to mention here but all deserve the thanks of all Samba users. To see a full list, look at the change-log in the source package for the pre-CVS changes and at <http://cvs.samba.org/> for the contributors to Samba post-CVS. CVS is the Open Source source code control system used by the Samba Team to develop Samba. The project would have been unmanageable without it.

Author

The original Samba software and related utilities were created by Andrew Tridgell. Samba is now developed by the Samba Team as an Open Source project similar to the way the

Linux kernel is developed.

The original Samba man pages were written by Karl Auer. The man page sources were converted to YODL format (another excellent piece of Open Source software, available at <ftp://ftp.icce.rug.nl/pub/unix/>) and updated for the Samba 2.0 release by Jeremy Allison. The conversion to DocBook for Samba 2.2 was done by Gerald Carter. The conversion to DocBook XML 4.2 for Samba 3.0 was done by Alexander Bokovoy.

Referenced By

[libsmbclient](#)(7), [lmhosts](#)(5), [nmblookup4](#)(1), [pam_winbind](#)(7), [pam_winbind](#)(8), [pam_winbind.conf](#)(5), [pmdasamba](#)(1), [sharesec](#)(1), [smbget](#)(1), [smbgetrc](#)(5), [smbpasswd](#)(5), [umount.cifs](#)(8), [vfs_acl_tdb](#)(8), [vfs_acl_xattr](#)(8), [vfs_aio_fork](#)(8), [vfs_aio_linux](#)(8), [vfs_aio_pthread](#)(8), [vfs_audit](#)(8), [vfs_cacheprime](#)(8), [vfs_cap](#)(8), [vfs_catia](#)(8), [vfs_commit](#)(8), [vfs_crossrename](#)(8), [vfs_default_quota](#)(8), [vfs_dirsort](#)(8), [vfs_extd_audit](#)(8), [vfs_fake_perms](#)(8), [vfs_fileid](#)(8), [vfs_full_audit](#)(8), [vfs_gpfs](#)(8), [vfs_media_harmony](#)(8), [vfs_netatalk](#)(8), [vfs_notify_fam](#)(8), [vfs_prealloc](#)(8), [vfs_preopen](#)(8), [vfs_readahead](#)(8), [vfs_READONLY](#)(8), [vfs_recycle](#)(8), [vfs_shadow_copy](#)(8), [vfs_shadow_copy2](#)(8), [vfs_smb_traffic_analyzer](#)(8), [vfs_streams_depot](#)(8), [vfs_streams_xattr](#)(8), [vfs_time_audit](#)(8), [vfs_tsmsm](#)(8), [vfs_xattr_tdb](#)(8), [winbind_krb5_locator](#)(7)

NAME

scp — OpenSSH secure file copy

SYNOPSIS

```
scp [ -346ABCOpqRrsTv ] [-c cipher] [-D sftp_server_path] [-F ssh_config]
[ -i identity_file ] [-J destination] [-l limit] [-o ssh_option] [-P port]
[ -S program] source ... target
```

DESCRIPTION

scp copies files between hosts on a network.

It uses **ssh(1)** for data transfer, and uses the same authentication and provides the same security as a login session.

scp will ask for passwords or passphrases if they are needed for authentication.

The *source* and *target* may be specified as a local pathname, a remote host with optional path in the form `[user@]host:[path]`, or a URI in the form `scp://[user@]host[:port][/path]`. Local file names can be made explicit using absolute or relative pathnames to avoid **scp** treating file names containing ‘`:`’ as host specifiers.

When copying between two remote hosts, if the URI format is used, a *port* cannot be specified on the *target* if the **-R** option is used.

The options are as follows:

- 3** Copies between two remote hosts are transferred through the local host. Without this option the data is copied directly between the two remote hosts. Note that, when using the original SCP protocol (the default), this option selects batch mode for the second host as **scp** cannot ask for passwords or passphrases for both hosts. This mode is the default.
- 4** Forces **scp** to use IPv4 addresses only.
- 6** Forces **scp** to use IPv6 addresses only.
- A** Allows forwarding of **ssh-agent(1)** to the remote system. The default is not to forward an authentication agent.
- B** Selects batch mode (prevents asking for passwords or passphrases).
- C** Compression enable. Passes the **-C** flag to **ssh(1)** to enable compression.
- c cipher**
Selects the cipher to use for encrypting the data transfer. This option is directly passed to **ssh(1)**.
- D sftp_server_path**
When using the SFTP protocol support via **-s**, connect directly to a local SFTP server program rather than a remote one via **ssh(1)**. This option may be useful in debugging the client and server.
- F ssh_config**
Specifies an alternative per-user configuration file for **ssh**. This option is directly passed to **ssh(1)**.
- i identity_file**
Selects the file from which the identity (private key) for public key authentication is read. This option is directly passed to **ssh(1)**.
- J destination**
Connect to the target host by first making an **scp** connection to the jump host described by *destination* and then establishing a TCP forwarding to the ultimate destination from there. Multiple jump hops may be specified separated by comma characters. This is a shortcut to specify a

ProxyJump configuration directive. This option is directly passed to **ssh(1)**.

-l *limit*

Limits the used bandwidth, specified in Kbit/s.

- o** Use the original SCP protocol for file transfers instead of the SFTP protocol. Forcing the use of the SCP protocol may be necessary for servers that do not implement SFTP, for backwards-compatibility for particular filename wildcard patterns and for expanding paths with a ‘~’ prefix for older SFTP servers. This mode is the default.

-o *ssh_option*

Can be used to pass options to **ssh** in the format used in **ssh_config(5)**. This is useful for specifying options for which there is no separate **scp** command-line flag. For full details of the options listed below, and their possible values, see **ssh_config(5)**.

AddressFamily
BatchMode
BindAddress
BindInterface
CanonicalDomains
CanonicalizeFallbackLocal
CanonicalizeHostname
CanonicalizeMaxDots
CanonicalizePermittedCNAMES
CASignatureAlgorithms
CertificateFile
CheckHostIP
Ciphers
Compression
ConnectionAttempts
ConnectTimeout
ControlMaster
ControlPath
ControlPersist
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIDelegateCredentials
HashKnownHosts
Host
HostbasedAcceptedAlgorithms
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
Hostname
IdentitiesOnly
IdentityAgent
IdentityFile
IPQoS
KbdInteractiveAuthentication
KbdInteractiveDevices
KexAlgorithms

KnownHostsCommand
LogLevel
MACs
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
PKCS11Provider
Port
PreferredAuthentications
ProxyCommand
ProxyJump
PubkeyAcceptedAlgorithms
PubkeyAuthentication
RekeyLimit
SendEnv
ServerAliveInterval
ServerAliveCountMax
SetEnv
StrictHostKeyChecking
TCPKeepAlive
UpdateHostKeys
User
UserKnownHostsFile
VerifyHostKeyDNS

-P *port*

Specifies the port to connect to on the remote host. Note that this option is written with a capital ‘P’, because **-p** is already reserved for preserving the times and mode bits of the file.

-p Preserves modification times, access times, and file mode bits from the source file.

-q Quiet mode: disables the progress meter as well as warning and diagnostic messages from **ssh(1)**.

-R Copies between two remote hosts are performed by connecting to the origin host and executing **scp** there. This requires that **scp** running on the origin host can authenticate to the destination host without requiring a password.

-r Recursively copy entire directories. Note that **scp** follows symbolic links encountered in the tree traversal.

-s *program*

Name of *program* to use for the encrypted connection. The program must understand **ssh(1)** options.

-s Use the SFTP protocol for transfers rather than the original scp protocol.

-T Disable strict filename checking. By default when copying files from a remote host to a local directory **scp** checks that the received filenames match those requested on the command-line to prevent the remote end from sending unexpected or unwanted files. Because of differences in how various operating systems and shells interpret filename wildcards, these checks may cause wanted files to be rejected. This option disables these checks at the expense of fully trusting that the server will not send unexpected filenames.

-v Verbose mode. Causes **scp** and **ssh(1)** to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.

EXIT STATUS

The **scp** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

sftp(1), **ssh(1)**, **ssh-add(1)**, **ssh-agent(1)**, **ssh-keygen(1)**, **ssh_config(5)**,
sftp-server(8), **sshd(8)**

HISTORY

scp is based on the **rcp** program in BSD source code from the Regents of the University of California.

AUTHORS

Timo Rinne <tri@iki.fi>
Tatu Ylonen <ylo@cs.hut.fi>

CAVEATS

The original SCP protocol (used by default) requires execution of the remote user's shell to perform **glob(3)** pattern matching. This requires careful quoting of any characters that have special meaning to the remote shell, such as quote characters.

NAME

sed – stream editor for filtering and transforming text

SYNOPSIS

sed [*OPTION*]... {*script-only-if-no-other-script*} [*input-file*]...

DESCRIPTION

Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as *ed*), *sed* works by making only one pass over the input(s), and is consequently more efficient. But it is *sed*'s ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

-n, --quiet, --silent

suppress automatic printing of pattern space

--debug

annotate program execution

-e script, --expression=script

add the script to the commands to be executed

-f script-file, --file=script-file

add the contents of script-file to the commands to be executed

--follow-symlinks

follow symlinks when processing in place

-i[SUFFIX], --in-place[=SUFFIX]

edit files in place (makes backup if SUFFIX supplied)

-l N, --line-length=N

specify the desired line-wrap length for the 'l' command

--posix

disable all GNU extensions.

-E, -r, --regexp-extended

use extended regular expressions in the script (for portability use POSIX -E).

-s, --separate

consider files as separate rather than as a single, continuous long stream.

--sandbox

operate in sandbox mode (disable e/r/w commands).

-u, --unbuffered

load minimal amounts of data from the input files and flush the output buffers more often

-z, --null-data

separate lines by NUL characters

--help

display this help and exit

--version

output version information and exit

If no **-e**, **--expression**, **-f**, or **--file** option is given, then the first non-option argument is taken as the *sed* script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.

GNU sed home page: <<https://www.gnu.org/software/sed/>>. General help using GNU software: <<https://www.gnu.org/gethelp/>>. E-mail bug reports to: <bug-sed@gnu.org>.

COMMAND SYNOPSIS

This is just a brief synopsis of *sed* commands to serve as a reminder to those who already know *sed*; other documentation (such as the texinfo document) must be consulted for fuller descriptions.

Zero-address “commands”

:label Label for **b** and **t** commands.

#comment

The comment extends until the next newline (or the end of a **-e** script fragment).

} The closing bracket of a { } block.

Zero- or One- address commands

= Print the current line number.

**a **

text Append *text*, which has each embedded newline preceded by a backslash.

**i **

text Insert *text*, which has each embedded newline preceded by a backslash.

q [exit-code]

Immediately quit the *sed* script without processing any more input, except that if auto-print is not disabled the current pattern space will be printed. The exit code argument is a GNU extension.

Q [exit-code]

Immediately quit the *sed* script without processing any more input. This is a GNU extension.

r filename

Append text read from *filename*.

R filename

Append a line read from *filename*. Each invocation of the command reads a line from the file. This is a GNU extension.

Commands which accept address ranges

{ Begin a block of commands (end with a **}**).

b label Branch to *label*; if *label* is omitted, branch to end of script.

**c **

text Replace the selected lines with *text*, which has each embedded newline preceded by a backslash.

d Delete pattern space. Start next cycle.

D If pattern space contains no newline, start a normal new cycle as if the **d** command was issued. Otherwise, delete text in the pattern space up to the first newline, and restart cycle with the resultant pattern space, without reading a new line of input.

h H Copy/append pattern space to hold space.

g G Copy/append hold space to pattern space.

l List out the current line in a “visually unambiguous” form.

l width List out the current line in a “visually unambiguous” form, breaking it at *width* characters. This is a GNU extension.

n N Read/append the next line of input into the pattern space.

p Print the current pattern space.

P Print up to the first embedded newline of the current pattern space.

s/regexp/replacement/

Attempt to match *regexp* against the pattern space. If successful, replace that portion matched with *replacement*. The *replacement* may contain the special character **&** to refer to that portion of the pattern space which matched, and the special escapes **\1** through **\9** to refer to the corresponding matching sub-expressions in the *regexp*.

t label If a **s///** has done a successful substitution since the last input line was read and since the last **t** or **T** command, then branch to *label*; if *label* is omitted, branch to end of script.

T label If no **s///** has done a successful substitution since the last input line was read and since the last **t** or **T** command, then branch to *label*; if *label* is omitted, branch to end of script. This is a GNU extension.

w filename

Write the current pattern space to *filename*.

W filename

Write the first line of the current pattern space to *filename*. This is a GNU extension.

x Exchange the contents of the hold and pattern spaces.

y/source/dest/

Transliterate the characters in the pattern space which appear in *source* to the corresponding character in *dest*.

Addresses

Sed commands can be given with no addresses, in which case the command will be executed for all input lines; with one address, in which case the command will only be executed for input lines which match that address; or with two addresses, in which case the command will be executed for all input lines which match the inclusive range of lines starting from the first address and continuing to the second address. Three things to note about address ranges: the syntax is *addr1,addr2* (i.e., the addresses are separated by a comma); the line which *addr1* matched will always be accepted, even if *addr2* selects an earlier line; and if *addr2* is a *regexp*, it will not be tested against the line that *addr1* matched.

After the address (or address-range), and before the command, a **!** may be inserted, which specifies that the command shall only be executed if the address (or address-range) does **not** match.

The following address types are supported:

number

Match only the specified line *number* (which increments cumulatively across files, unless the **-s** option is specified on the command line).

first~step

Match every *step*'th line starting with line *first*. For example, “*sed -n 1~2p*” will print all the odd-numbered lines in the input stream, and the address *2~5* will match every fifth line, starting with the second. *first* can be zero; in this case, *sed* operates as if it were equal to *step*. (This is an extension.)

\$ Match the last line.

/regexp/

Match lines matching the regular expression *regexp*. Matching is performed on the current pattern space, which can be modified with commands such as “**s///**”.

\cregexp\c

Match lines matching the regular expression *regexp*. The **c** may be an **y** character.

GNU *sed* also supports some special 2-address forms:

0,addr2

Start out in "matched first address" state, until *addr2* is found. This is similar to *1,addr2*, except that if *addr2* matches the very first line of input the *0,addr2* form will be at the end of its range, whereas the *1,addr2* form will still be at the beginning of its range. This works only when *addr2*

is a regular expression.

addr1,+N

Will match *addr1* and the *N* lines following *addr1*.

addr1,~N

Will match *addr1* and the lines following *addr1* until the next line whose input line number is a multiple of *N*.

REGULAR EXPRESSIONS

POSIX.2 BREs *should* be supported, but they aren't completely because of performance problems. The **\n** sequence in a regular expression matches the newline character, and similarly for **\a**, **\t**, and other sequences. The **-E** option switches to using extended regular expressions instead; it has been supported for years by GNU sed, and is now included in POSIX.

BUGS

E-mail bug reports to bug-sed@gnu.org. Also, please include the output of “sed --version” in the body of your report if at all possible.

AUTHOR

Written by Jay Fenlason, Tom Lord, Ken Pizzini, Paolo Bonzini, Jim Meyering, and Assaf Gordon.

This sed program was built with SELinux support. SELinux is enabled on this system.

GNU sed home page: <<https://www.gnu.org/software/sed/>>. General help using GNU software: <<https://www.gnu.org/gethelp/>>. E-mail bug reports to: <bug-sed@gnu.org>.

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

awk(1), **ed(1)**, **grep(1)**, **tr(1)**, **perlre(1)**, **sed.info**, any of various books on **sed**, the **sed** FAQ (<http://sed.sf.net/grabbag/tutorials/sedfaq.txt>), <http://sed.sf.net/grabbag/>.

The full documentation for **sed** is maintained as a Texinfo manual. If the **info** and **sed** programs are properly installed at your site, the command

info sed

should give you access to the complete manual.

NAME

service – run a System V init script

SYNOPSIS

service *SCRIPT COMMAND [OPTIONS]*

service --status-all

service --help | -h | --version

DESCRIPTION

service runs a System V init script or systemd unit in as predictable an environment as possible, removing most environment variables and with the current working directory set to */*.

The *SCRIPT* parameter specifies a System V init script, located in */etc/init.d/SCRIPT*, or the name of a systemd unit. The existence of a systemd unit of the same name as a script in */etc/init.d* will cause the unit to take precedence over the init.d script. The supported values of *COMMAND* depend on the invoked script. **service** passes *COMMAND* and *OPTIONS* to the init script unmodified. For systemd units, start, stop, status, and reload are passed through to their systemctl/initctl equivalents.

All scripts should support at least the **start** and **stop** commands. As a special case, if *COMMAND* is **--full-restart**, the script is run twice, first with the **stop** command, then with the **start** command. Note, that unlike **update-rc.d(8)**, **service** does not check */usr/sbin/policy-rc.d*.

service --status-all runs all init scripts, in alphabetical order, with the **status** command. The status is [+] for running services, [-] for stopped services and [?] for services without **astatus** command. This option only calls status for sysvinit jobs.

EXIT CODES

service calls the init script and returns the status returned by it.

FILES

/etc/init.d

The directory containing System V init scripts.

{lib,run,etc}/systemd/system

The directories containing systemd units.

ENVIRONMENT

LANG, LANGUAGE, LC_CTYPE, LC_NUMERIC, LC_TIME, LC_COLLATE, LC_MONETARY, LC_MESSAGES, LC_PAPER, LC_NAME, LC_ADDRESS, LC_TELEPHONE, LC_MEASUREMENT, LC_IDENTIFICATION, LC_ALL, TERM, PATH

The only environment variables passed to the init scripts.

SEE ALSO

/etc/init.d/skeleton

update-rc.d(8)

init(8)

invoke-rc.d(8)

systemctl(1)

AUTHOR

Miloslav Trmac <mitr@redhat.com>, Petter Reinholdtsen <pere@hungry.com>

License: GNU General Public License v2 (GPLv2)

COPYRIGHT

2006 Red Hat, Inc., Petter Reinholdtsen <pere@hungry.com>

SERVICE(8)

System Manager's Manual

SERVICE(8)

NAME

services – Internet network services list

DESCRIPTION

services is a plain ASCII file providing a mapping between human-friendly textual names for internet services, and their underlying assigned port numbers and protocol types. Every networking program should look into this file to get the port number (and protocol) for its service. The C library routines **getservent(3)**, **getservbyname(3)**, **getservbyport(3)**, **setservent(3)**, and **endservent(3)** support querying this file from programs.

Port numbers are assigned by the IANA (Internet Assigned Numbers Authority), and their current policy is to assign both TCP and UDP protocols when assigning a port number. Therefore, most entries will have two entries, even for TCP-only services.

Port numbers below 1024 (so-called "low numbered" ports) can be bound to only by root (see **bind(2)**, **tcp(7)**, and **udp(7)**). This is so clients connecting to low numbered ports can trust that the service running on the port is the standard implementation, and not a rogue service run by a user of the machine. Well-known port numbers specified by the IANA are normally located in this root-only space.

The presence of an entry for a service in the **services** file does not necessarily mean that the service is currently running on the machine. See **inetd.conf(5)** for the configuration of Internet services offered. Note that not all networking services are started by **inetd(8)**, and so won't appear in **inetd.conf(5)**. In particular, news (NNTP) and mail (SMTP) servers are often initialized from the system boot scripts.

The location of the **services** file is defined by **_PATH_SERVICES** in **<netdb.h>**. This is usually set to **/etc/services**.

Each line describes one service, and is of the form:

```
service-name port/protocol [aliases ...]
```

where:

service-name

is the friendly name the service is known by and looked up under. It is case sensitive. Often, the client program is named after the *service-name*.

port is the port number (in decimal) to use for this service.

protocol

is the type of protocol to be used. This field should match an entry in the **protocols(5)** file. Typical values include **tcp** and **udp**.

aliases is an optional space or tab separated list of other names for this service. Again, the names are case sensitive.

Either spaces or tabs may be used to separate the fields.

Comments are started by the hash sign (#) and continue until the end of the line. Blank lines are skipped.

The *service-name* should begin in the first column of the file, since leading spaces are not stripped. *service-names* can be any printable characters excluding space and tab. However, a conservative choice of characters should be used to minimize compatibility problems. For example, a–z, 0–9, and hyphen (–) would seem a sensible choice.

Lines not matching this format should not be present in the file. (Currently, they are silently skipped by **getservent(3)**, **getservbyname(3)**, and **getservbyport(3)**. However, this behavior should not be relied on.)

This file might be distributed over a network using a network-wide naming service like Yellow Pages/NIS or BIND/Hesiod.

A sample **services** file might look like this:

netstat	15/tcp	
qotd	17/tcp	quote
msp	18/tcp	# message send protocol

```
msp          18/udp      # message send protocol
chargen     19/tcp       ttytst source
chargen     19/udp      ttytst source
ftp          21/tcp
# 22 - unassigned
telnet      23/tcp
```

FILES

/etc/services

The Internet network services list

<netdb.h>

Definition of **_PATH_SERVICES**

SEE ALSO

listen(2), endservent(3), getservbyname(3), getservbyport(3), getservent(3), setservent(3), inetd.conf(5), protocols(5), inetd(8)

Assigned Numbers RFC, most recently RFC 1700, (AKA STD0002).

NAME

setfacl – set file access control lists

SYNOPSIS

```
setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...  
setfacl --restore={file|-}
```

DESCRIPTION

This utility sets Access Control Lists (ACLs) of files and directories. On the command line, a sequence of commands is followed by a sequence of files (which in turn can be followed by another sequence of commands, ...).

The *-m* and *-x* options expect an ACL on the command line. Multiple ACL entries are separated by comma characters (','). The *-M* and *-X* options read an ACL from a file or from standard input. The ACL entry format is described in Section ACL ENTRIES.

The *--set* and *--set-file* options set the ACL of a file or a directory. The previous ACL is replaced. ACL entries for this operation must include permissions.

The *-m* (*--modify*) and *-M* (*--modify-file*) options modify the ACL of a file or directory. ACL entries for this operation must include permissions.

The *-x* (*--remove*) and *-X* (*--remove-file*) options remove ACL entries. It is not an error to remove an entry which does not exist. Only ACL entries without the *perms* field are accepted as parameters, unless *POSIXLY_CORRECT* is defined.

When reading from files using the *-M* and *-X* options, *setf acl* accepts the output *getfacl* produces. There is at most one ACL entry per line. After a Pound sign ('#'), everything up to the end of the line is treated as a comment.

If *setfacl* is used on a file system which does not support ACLs, *setfacl* operates on the file mode permission bits. If the ACL does not fit completely in the permission bits, *setfacl* modifies the file mode permission bits to reflect the ACL as closely as possible, writes an error message to standard error, and returns with an exit status greater than 0.

PERMISSIONS

The file owner and processes capable of *CAP_FOWNER* are granted the right to modify ACLs of a file. This is analogous to the permissions required for accessing the file mode. (On current Linux systems, root is the only user with the *CAP_FOWNER* capability.)

OPTIONS

- b, --remove-all*
Remove all extended ACL entries. The base ACL entries of the owner, group and others are retained.
- k, --remove-default*
Remove the Default ACL. If no Default ACL exists, no warnings are issued.
- n, --no-mask*
Do not recalculate the effective rights mask. The default behavior of *setfacl* is to recalculate the ACL mask entry, unless a mask entry was explicitly given. The mask entry is set to the union of all permissions of the owning group, and all named user and group entries. (These are exactly the entries affected by the mask entry).

--mask

Do recalculate the effective rights mask, even if an ACL mask entry was explicitly given. (See the **-n** option.)

-d, --default

All operations apply to the Default ACL. Regular ACL entries in the input set are promoted to Default ACL entries. Default ACL entries in the input set are discarded. (A warning is issued if that happens).

--restore={file}/-

Restore a permission backup created by ‘getfacl -R’ or similar. All permissions of a complete directory subtree are restored using this mechanism. If the input contains owner comments or group comments, setfacl attempts to restore the owner and owning group. If the input contains flags comments (which define the setuid, setgid, and sticky bits), setfacl sets those three bits accordingly; otherwise, it clears them. This option cannot be mixed with other options except ‘--test’. If the file specified is ‘-’, then it will be read from standard input.

--test

Test mode. Instead of changing the ACLs of any files, the resulting ACLs are listed.

-R, --recursive

Apply operations to all files and directories recursively. This option cannot be mixed with ‘--restore’.

-L, --logical

Logical walk, follow symbolic links to directories. The default behavior is to follow symbolic link arguments, and skip symbolic links encountered in subdirectories. Only effective in combination with **-R**. This option cannot be mixed with ‘--restore’.

-P, --physical

Physical walk, do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with **-R**. This option cannot be mixed with ‘--restore’.

-v, --version

Print the version of setfacl and exit.

-h, --help

Print help explaining the command line options.

-- End of command line options. All remaining parameters are interpreted as file names, even if they start with a dash.

– If the file name parameter is a single dash, setfacl reads a list of files from standard input.

ACL ENTRIES

The setfacl utility recognizes the following ACL entry formats (blanks inserted for clarity):

[d[efault]:] [u[ser]:]uid [:perms]

Permissions of a named user. Permissions of the file owner if *uid* is empty.

[d[efault]:] g[roup]:gid [:perms]

Permissions of a named group. Permissions of the owning group if *gid* is empty.

[d[efault]:] m[ask][:] [:perms]

Effective rights mask

[d[efault]:] o[ther][:] [:perms]

Permissions of others.

Whitespace between delimiter characters and non-delimiter characters is ignored.

Proper ACL entries including permissions are used in modify and set operations. (options **-m**, **-M**, **--set** and **--set-file**). Entries without the *perms* field are used for *deletion* of entries (options **-x** and **-X**).

For *uid* and *gid* you can specify either a name or a number. Character literals may be specified with a

backslash followed by the 3-digit octal digits corresponding to the ASCII code for the character (e.g., \101 for 'A'). If the name contains a literal backslash followed by 3 digits, the backslash must be escaped (i.e., \\).

The *perms* field is a combination of characters that indicate the read (*r*), write (*w*), execute (*x*) permissions. Dash characters in the *perms* field (–) are ignored. The character *X* stands for the execute permission if the file is a directory or already has execute permission for some user. Alternatively, the *perms* field can define the permissions numerically, as a bit-wise combination of read (4), write (2), and execute (1). Zero *perms* fields or *perms* fields that only consist of dashes indicate no permissions.

AUTOMATICALLY CREATED ENTRIES

Initially, files and directories contain only the three base ACL entries for the owner, the group, and others. There are some rules that need to be satisfied in order for an ACL to be valid:

- * The three base entries cannot be removed. There must be exactly one entry of each of these base entry types.
- * Whenever an ACL contains named user entries or named group objects, it must also contain an effective rights mask.
- * Whenever an ACL contains any Default ACL entries, the three Default ACL base entries (default owner, default group, and default others) must also exist.
- * Whenever a Default ACL contains named user entries or named group objects, it must also contain a default effective rights mask.

To help the user ensure these rules, setfac1 creates entries from existing entries under the following conditions:

- * If an ACL contains named user or named group entries, and no mask entry exists, a mask entry containing the same permissions as the group entry is created. Unless the *-n* option is given, the permissions of the mask entry are further adjusted to include the union of all permissions affected by the mask entry. (See the *-n* option description).
- * If a Default ACL entry is created, and the Default ACL contains no owner, owning group, or others entry, a copy of the ACL owner, owning group, or others entry is added to the Default ACL.
- * If a Default ACL contains named user entries or named group entries, and no mask entry exists, a mask entry containing the same permissions as the default Default ACL's group entry is added. Unless the *-n* option is given, the permissions of the mask entry are further adjusted to include the union of all permissions affected by the mask entry. (See the *-n* option description).

EXAMPLES

Granting an additional user read access

```
setfac1 -m u:lisa:r file
```

Revoking write access from all groups and all named users (using the effective rights mask)

```
setfac1 -m m:::rx file
```

Removing a named group entry from a file's ACL

```
setfac1 -x g:staff file
```

Copying the ACL of one file to another

```
getfac1 file1 | setfac1 --set-file=- file2
```

Copying the access ACL into the Default ACL

```
getfac1 --access dir | setfac1 -d -M- dir
```

CONFORMANCE TO POSIX 1003.1e DRAFT STANDARD 17

If the environment variable `POSIXLY_CORRECT` is defined, the default behavior of `setfac1` changes as follows: All non-standard options are disabled. The "default:" prefix is disabled. The `-x` and `-X` options also accept permission fields (and ignore them).

AUTHOR

Andreas Gruenbacher, <*andreas.gruenbacher@gmail.com*>.

Please send your bug reports, suggested features and comments to the above address.

SEE ALSO

getfacl(1), chmod(1), umask(1), acl(5)

NAME

setfattr – set extended attributes of filesystem objects

SYNOPSIS

```
setfattr [-h] -n name [-v value] pathname...
setfattr [-h] -x name pathname...
setfattr [-h] --restore=file
```

DESCRIPTION

The **setfattr** command associates a new *value* with an extended attribute *name* for each specified file.

OPTIONS

-n *name*, **--name**=*name*

Specifies the name of the extended attribute to set.

-v *value*, **--value**=*value*

Specifies the new value of the extended attribute. There are three methods available for encoding the value. If the given string is enclosed in double quotes, the inner string is treated as text. In that case, backslashes and double quotes have special meanings and need to be escaped by a preceding backslash. Any control characters can be encoded as a backslash followed by three digits as its ASCII code in octal. If the given string begins with 0x or 0X, it expresses a hexadecimal number. If the given string begins with 0s or 0S, base64 encoding is expected. Also see the **--encoding** option of **getattr(1)**.

-x *name*, **--remove**=*name*

Remove the named extended attribute entirely.

-h, **--no-dereference**

Do not follow symlinks. If *pathname* is a symbolic link, it is not followed, but is instead itself the inode being modified.

--restore=*file*

Restores extended attributes from file. The file must be in the format generated by the **getattr** command with the **--dump** option. If a dash (–) is given as the file name, **setfattr** reads from standard input.

--raw

Do not decode the attribute value. Can be used to set values obtained with **getattr --only-values**.

--version

Print the version of **setfattr** and exit.

--help

Print help explaining the command line options.

-- End of command line options. All remaining parameters are interpreted as file names, even if they start with a dash character.

EXAMPLES

Add extended attribute to user namespace:

```
$ setfattr -n user.foo -v bar file.txt
```

To add md5sum of the file as an extended attribute:

```
# setfattr -n trusted.md5sum -v d41d8cd98f00b204e00998ecf8427e file.txt
```

AUTHOR

Andreas Gruenbacher, <andreas.gruenbacher@gmail.com> and the SGI XFS development team, <linux-xfs@oss.sgi.com>.

Please send your bug reports or comments to <<https://savannah.nongnu.org/bugs/?group=attr>> or <acl-devel@nongnu.org>.

SEE ALSO

getfattr(1), attr(5)

NAME

`gethostname`, `sethostname` – get/set hostname

LIBRARY

Standard C library (`libc`, `-lc`)

SYNOPSIS

```
#include <unistd.h>
int gethostname(char *name, size_t len);
int sethostname(const char *name, size_t len);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

```
gethostname():
_XOPEN_SOURCE >= 500 || _POSIX_C_SOURCE >= 200112L
|| /* glibc 2.19 and earlier */ _BSD_SOURCE

sethostname():
Since glibc 2.21:
_DEFAULT_SOURCE
In glibc 2.19 and 2.20:
_DEFAULT_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
Up to and including glibc 2.19:
_BSD_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
```

DESCRIPTION

These system calls are used to access or to change the system hostname. More precisely, they operate on the hostname associated with the calling process's UTS namespace.

sethostname() sets the hostname to the value given in the character array *name*. The *len* argument specifies the number of bytes in *name*. (Thus, *name* does not require a terminating null byte.)

gethostname() returns the null-terminated hostname in the character array *name*, which has a length of *len* bytes. If the null-terminated hostname is too large to fit, then the name is truncated, and no error is returned (but see NOTES below). POSIX.1 says that if such truncation occurs, then it is unspecified whether the returned buffer includes a terminating null byte.

RETURN VALUE

On success, zero is returned. On error, `-1` is returned, and *errno* is set to indicate the error.

ERRORS**FAULT**

name is an invalid address.

EINVAL

len is negative or, for **sethostname**(*len*), *len* is larger than the maximum allowed size.

ENAMETOOLONG

(glibc **gethostname**(*len*) *len* is smaller than the actual size. (Before glibc 2.1, glibc uses **EINVAL** for this case.)

EPERM

For **sethostname**(*len*), the caller did not have the **CAP_SYS_ADMIN** capability in the user namespace associated with its UTS namespace (see **namespaces**(7)).

STANDARDS

SVr4, 4.4BSD (these interfaces first appeared in 4.2BSD). POSIX.1-2001 and POSIX.1-2008 specify **gethostname**(*len*) but not **sethostname**(*len*).

NOTES

SUSv2 guarantees that "Host names are limited to 255 bytes". POSIX.1 guarantees that "Host names (not including the terminating null byte) are limited to **HOST_NAME_MAX** bytes". On Linux, **HOST_NAME_MAX** is defined with the value 64, which has been the limit since Linux 1.0 (earlier

kernels imposed a limit of 8 bytes).

C library/kernel differences

The GNU C library does not employ the **gethostname()** system call; instead, it implements **gethostname()** as a library function that calls **uname(2)** and copies up to *len* bytes from the returned *nodename* field into *name*. Having performed the copy, the function then checks if the length of the *nodename* was greater than or equal to *len*, and if it is, then the function returns -1 with *errno* set to **ENAMETOOLONG**; in this case, a terminating null byte is not included in the returned *name*.

Versions of glibc before glibc 2.2 handle the case where the length of the *nodename* was greater than or equal to *len* differently: nothing is copied into *name* and the function returns -1 with *errno* set to **ENAME-TOOLONG**.

SEE ALSO

hostname(1), **getdomainname(2)**, **setdomainname(2)**, **uname(2)**, **uts_namespaces(7)**

NAME

`setxattr`, `lsetxattr`, `fsetxattr` – set an extended attribute value

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/xattr.h>

int setxattr(const char *path, const char *name,
             const void value[.size], size_t size, int flags);
int lsetxattr(const char *path, const char *name,
              const void value[.size], size_t size, int flags);
int fsetxattr(int fd, const char *name,
              const void value[.size], size_t size, int flags);
```

DESCRIPTION

Extended attributes are *name:value* pairs associated with inodes (files, directories, symbolic links, etc.). They are extensions to the normal attributes which are associated with all inodes in the system (i.e., the `stat(2)` data). A complete overview of extended attributes concepts can be found in **xattr(7)**.

setxattr() sets the *value* of the extended attribute identified by *name* and associated with the given *path* in the filesystem. The *size* argument specifies the size (in bytes) of *value*; a zero-length value is permitted.

lsetxattr() is identical to **setxattr()**, except in the case of a symbolic link, where the extended attribute is set on the link itself, not the file that it refers to.

fsetxattr() is identical to **setxattr()**, only the extended attribute is set on the open file referred to by *fd* (as returned by `open(2)`) in place of *path*.

An extended attribute name is a null-terminated string. The *name* includes a namespace prefix; there may be several, disjoint namespaces associated with an individual inode. The *value* of an extended attribute is a chunk of arbitrary textual or binary data of specified length.

By default (i.e., *flags* is zero), the extended attribute will be created if it does not exist, or the value will be replaced if the attribute already exists. To modify these semantics, one of the following values can be specified in *flags*:

XATTR_CREATE

Perform a pure create, which fails if the named attribute exists already.

XATTR_REPLACE

Perform a pure replace operation, which fails if the named attribute does not already exist.

RETURN VALUE

On success, zero is returned. On failure, -1 is returned and *errno* is set to indicate the error.

ERRORS**EDQUOT**

Disk quota limits meant that there is insufficient space remaining to store the extended attribute.

EEXIST

XATTR_CREATE was specified, and the attribute exists already.

ENODATA

XATTR_REPLACE was specified, and the attribute does not exist.

ENOSPC

There is insufficient space remaining to store the extended attribute.

ENOTSUP

The namespace prefix of *name* is not valid.

ENOTSUP

Extended attributes are not supported by the filesystem, or are disabled,

EPERM

The file is marked immutable or append-only. (See [ioctl_iflags\(2\)](#).)

In addition, the errors documented in [stat\(2\)](#) can also occur.

ERANGE

The size of *name* or *value* exceeds a filesystem-specific limit.

VERSIONS

These system calls have been available since Linux 2.4; glibc support is provided since glibc 2.3.

STANDARDS

These system calls are Linux-specific.

SEE ALSO

[getattr\(1\)](#), [setattr\(1\)](#), [getxattr\(2\)](#), [listxattr\(2\)](#), [open\(2\)](#), [removexattr\(2\)](#), [stat\(2\)](#), [symlink\(7\)](#), [xattr\(7\)](#)

NAME

sfdisk – display or manipulate a disk partition table

SYNOPSIS

sfdisk [options] *device* [-N *partition-number*]

sfdisk [options] *command*

DESCRIPTION

sfdisk is a script-oriented tool for partitioning any block device. It runs in interactive mode if executed on a terminal (stdin refers to a terminal).

Since version 2.26 **sfdisk** supports MBR (DOS), GPT, SUN and SGI disk labels, but no longer provides any functionality for CHS (Cylinder–Head–Sector) addressing. CHS has never been important for Linux, and this addressing concept does not make any sense for new devices.

sfdisk protects the first disk sector when create a new disk label. The option **--wipe always** disables this protection. Note that **fdisk(8)** and **cfdisk(8)** completely erase this area by default.

sfdisk (since version 2.26) **aligns the start and end of partitions** to block-device I/O limits when relative sizes are specified, when the default values are used or when multiplicative suffixes (e.g., MiB) are used for sizes. It is possible that partition size will be optimized (reduced or enlarged) due to alignment if the start offset is specified exactly in sectors and partition size relative or by multiplicative suffixes.

The recommended way is not to specify start offsets at all and specify partition size in MiB, GiB (or so). In this case **sfdisk** aligns all partitions to block-device I/O limits (or when I/O limits are too small then to megabyte boundary to keep disk layout portable). If this default behaviour is unwanted (usually for very small partitions) then specify offsets and sizes in sectors. In this case **sfdisk** entirely follows specified numbers without any optimization.

sfdisk does not create the standard system partitions for SGI and SUN disk labels like **fdisk(8)** does. It is necessary to explicitly create all partitions including whole-disk system partitions.

sfdisk uses **BLKRRPART** (reread partition table) ioctl to make sure that the device is not used by system or other tools (see also **--no-reread**). It's possible that this feature or another **sfdisk** activity races with **systemd-udevd(8)**. The recommended way how to avoid possible collisions is to use **--lock** option. The exclusive lock will cause **systemd-udevd** to skip the event handling on the device.

The **sfdisk** prompt is only a hint for users and a displayed partition number does not mean that the same partition table entry will be created (if **-N** not specified), especially for tables with gaps.

COMMANDS

The commands are mutually exclusive.

[**-N** *partition-number*] *device*

The default **sfdisk** command is to read the specification for the desired partitioning of *device* from standard input, and then create a partition table according to the specification. See below for the description of the input format. If standard input is a terminal, then **sfdisk** starts an interactive session.

If the option **-N** is specified, then the changes are applied to the partition addressed by *partition-number*. The unspecified fields of the partition are not modified.

Note that it's possible to address an unused partition with **-N**. For example, an MBR always contains 4 partitions, but the number of used partitions may be smaller. In this case **sfdisk** follows the default values from the partition table and does not use built-in defaults for the unused partition given with **-N**. See also **--append**.

-A, --activate device [partition-number...]

Switch on the bootable flag for the specified partitions and switch off the bootable flag on all unspecified partitions. The special placeholder '-' may be used instead of the partition numbers to switch off the bootable flag on all partitions.

The activation command is supported for MBR and PMBR only. If a GPT label is detected, then **sfdisk** prints warning and automatically enters PMBR.

If no *partition-number* is specified, then list the partitions with an enabled flag.

--backup-pt-sectors device

Back up the current partition table sectors in binary format and exit. See the **BACKING UP THE PARTITION TABLE** section.

--delete device [partition-number...]

Delete all or the specified partitions.

-d, --dump device

Dump the partitions of a device in a format that is usable as input to **sfdisk**. See the **BACKING UP THE PARTITION TABLE** section.

-g, --show-geometry [device...]

List the geometry of all or the specified devices. For backward compatibility the deprecated option **--show-pt-geometry** have the same meaning as this one.

-J, --json device

Dump the partitions of a device in JSON format. Note that **sfdisk** is not able to use JSON as input format.

-l, --list [device...]

List the partitions of all or the specified devices. This command can be used together with **--verify**.

-F, --list-free [device...]

List the free unpartitioned areas on all or the specified devices.

--part-attrs device partition-number [attributes]

Change the GPT partition attribute bits. If *attributes* is not specified, then print the current partition settings. The *attributes* argument is a comma- or space-delimited list of bits numbers or bit names. For example, the string "RequiredPartition,50,51" sets three bits. The currently supported attribute bits are:

Bit 0 (RequiredPartition)

If this bit is set, the partition is required for the platform to function. The creator of the partition indicates that deletion or modification of the contents can result in loss of platform features or failure for the platform to boot or operate. The system cannot function normally if this partition is removed, and it should be considered part of the hardware of the system.

Bit 1 (NoBlockIOProtocol)

EFI firmware should ignore the content of the partition and not try to read from it.

Bit 2 (LegacyBIOSBootable)

The partition may be bootable by legacy BIOS firmware.

Bits 3–47

Undefined and must be zero. Reserved for expansion by future versions of the UEFI specification.

Bits 48–63

Reserved for GUID specific use. The use of these bits will vary depending on the partition type. For example Microsoft uses bit 60 to indicate read-only, 61 for shadow copy of another partition, 62 for hidden partitions and 63 to disable automount.

--part-label *device partition-number [label]*

Change the GPT partition name (label). If *label* is not specified, then print the current partition label.

--part-type *device partition-number [type]*

Change the partition type. If *type* is not specified, then print the current partition type.

The *type* argument is hexadecimal for MBR, GUID for GPT, type alias (e.g. "linux") or type shortcut (e.g. 'L'). For backward compatibility the options **-c** and **--id** have the same meaning as this one.

--part-uuid *device partition-number [uuid]*

Change the GPT partition UUID. If *uuid* is not specified, then print the current partition UUID.

--disk-id *device [id]*

Change the disk identifier. If *id* is not specified, then print the current identifier. The identifier is UUID for GPT or unsigned integer for MBR.

-r, --reorder *device*

Renumber the partitions, ordering them by their start offset.

-s, --show-size [*device...*]

List the sizes of all or the specified devices in units of 1024 byte size. This command is DEPRECATED in favour of **blockdev(8)**.

-T, --list-types

Print all supported types for the current disk label or the label specified by **--label**.

-V, --verify [*device...*]

Test whether the partition table and partitions seem correct.

--relocate *oper device*

Relocate partition table header. This command is currently supported for GPT header only. The argument *oper* can be:

gpt-bak-std

Move GPT backup header to the standard location at the end of the device.

gpt-bak-mini

Move GPT backup header behind the last partition. Note that UEFI standard requires the backup header at the end of the device and partitioning tools can automatically relocate the header to follow the standard.

OPTIONS

-a, --append

Don't create a new partition table, but only append the specified partitions.

Note that unused partition maybe be re-used in this case although it is not the last partition in the partition table. See also **-N** to specify entry in the partition table.

-b, --backup

Back up the current partition table sectors before starting the partitioning. The default backup file name is `/sfdisk-<device>-<offset>.bak`; to use another name see option **-O, --backup-file**. See section **BACKING UP THE PARTITION TABLE** for more details.

--color[=when]

Colorize the output. The optional argument *when* can be **auto**, **never** or **always**. If the *when* argument is omitted, it defaults to **auto**. The colors can be disabled; for the current built-in default see the **--help** output. See also the **COLORS** section.

-f, --force

Disable all consistency checking.

--Linux

Deprecated and ignored option. Partitioning that is compatible with Linux (and other modern operating systems) is the default.

--lock[=mode]

Use exclusive BSD lock for device or file it operates. The optional argument *mode* can be **yes**, **no** (or 1 and 0) or **nonblock**. If the *mode* argument is omitted, it defaults to **yes**. This option overwrites environment variable `$LOCK_BLOCK_DEVICE`. The default is not to use any lock at all, but it's recommended to avoid collisions with **systemd-udevd(8)** or other tools.

-n, --no-act

Do everything except writing to the device.

--no-reread

Do not check through the re-read-partition-table ioctl whether the device is in use.

--no-tell-kernel

Don't tell the kernel about partition changes. This option is recommended together with **--no-reread** to modify a partition on used disk. The modified partition should not be used (e.g., mounted).

-O, --backup-file path

Override the default backup file name. Note that the device name and offset are always appended to the file name.

--move-data[=path]

Move data after partition relocation, for example when moving the beginning of a partition to another place on the disk. The size of the partition has to remain the same, the new and old location may overlap. This option requires option **-N** in order to be processed on one specific partition only.

The optional *path* specifies log file name. The log file contains information about all read/write operations on the partition data. The word "@default" as a *path* forces **sfdisk** to use `/sfdisk-<devname>.move` for the log. The log is optional since v2.35.

Note that this operation is risky and not atomic. **Don't forget to backup your data!**

See also **--move-use-fsync**.

In the example below, the first command creates a 100MiB free area before the first partition and moves the data it contains (e.g., a filesystem), the next command creates a new partition from the free space (at offset 2048), and the last command reorders partitions to match disk order (the original sdc1 will become sdc2).

```
echo '+100M,' | sfdisk --move-data /dev/sdc -N 1 echo '2048,' | sfdisk /dev/sdc --append sfdisk /dev/sdc --reorder
```

--move-use-fsync

Use the **fsync**(2) system call after each write when moving data to a new location by **--move-data**.

-o, --output list

Specify which output columns to print. Use **--help** to get a list of all supported columns.

The default list of columns may be extended if *list* is specified in the format *+list* (e.g., **-o +UUID**).

-q, --quiet

Suppress extra info messages.

-u, --unit S

Deprecated option. Only the sector unit is supported. This option is not supported when using the **--show-size** command.

-X, --label type

Specify the disk label type (e.g., **dos**, **gpt**, ...). If this option is not given, then **sfdisk** defaults to the existing label, but if there is no label on the device yet, then the type defaults to **dos**. The default or the current label may be overwritten by the "label: <name>" script header line. The option **--label** does not force **sfdisk** to create empty disk label (see the **EMPTY DISK LABEL** section below).

-Y, --label-nested type

Force editing of a nested disk label. The primary disk label has to exist already. This option allows editing for example a hybrid/protective MBR on devices with GPT.

-w, --wipe when

Wipe filesystem, RAID and partition-table signatures from the device, in order to avoid possible collisions. The argument *when* can be **auto**, **never** or **always**. When this option is not given, the default is **auto**, in which case signatures are wiped only when in interactive mode; except the old partition-table signatures which are always wiped before create a new partition-table if the argument *when* is not **never**. The **auto** mode also does not wipe the first sector (boot sector), it is necessary to use the **always** mode to wipe this area. In all cases detected signatures are reported by warning messages before a new partition table is created. See also the **wipefs(8)** command.

-W, --wipe-partitions when

Wipe filesystem, RAID and partition-table signatures from a newly created partition, in order to avoid possible collisions. The argument *when* can be **auto**, **never** or **always**. When this option is not given, the default is **auto**, in which case signatures are wiped only when in interactive mode and after confirmation by user. In all cases detected signatures are reported by warning messages after a new partition is created. See also **wipefs(8)** command.

-v, --version

Display version information and exit.

-h, --help

Display help text and exit.

INPUT FORMATS

sfdisk supports two input formats and generic header lines.

Header lines

The optional header lines specify generic information that apply to the partition table. The header-line format is:

<name>: <value>

The currently recognized headers are:

unit

Specify the partitioning unit. The only supported unit is **sectors**.

label

Specify the partition table type. For example **dos** or **gpt**.

label-id

Specify the partition table identifier. It should be a hexadecimal number (with a 0x prefix) for MBR and a UUID for GPT.

first-lba

Specify the first usable sector for GPT partitions.

last-lba

Specify the last usable sector for GPT partitions.

table-length

Specify the maximal number of GPT partitions.

grain

Specify minimal size in bytes used to calculate partitions alignment. The default is 1MiB and it's strongly recommended to use the default. Do not modify this variable if you're not sure.

sector-size

Specify sector size. This header is informative only and it is not used when **sfdisk** creates a new partition table, in this case the real device specific value is always used and sector size from the dump is ignored.

Note that it is only possible to use header lines before the first partition is specified in the input.

Unnamed-fields format

start size type bootable

where each line fills one partition descriptor.

Fields are separated by whitespace, comma (recommended) or semicolon possibly followed by whitespace; initial and trailing whitespace is ignored. Numbers can be octal, decimal or hexadecimal; decimal is the default. When a field is absent, empty or specified as '-' a default value is used. But when the **-N** option (change a single partition) is given, the default for each field is its previous value.

The default value of *start* is the first non-assigned sector aligned according to device I/O limits. The default start offset for the first partition is 1 MiB. If the offset is followed by the multiplicative suffixes (KiB, MiB, GiB, TiB, PiB, EiB, ZiB and YiB), then the number is interpreted as offset in bytes. Since v2.38 when the **-N** option (change a single partition) is given, a '+' can be used to enlarge partition by move start of the partition if there is a free space before the partition.

The default value of *size* indicates "as much as possible"; i.e., until the next partition or end-of-device. A numerical argument is by default interpreted as a number of sectors, however if the size is followed by one of the multiplicative suffixes (KiB, MiB, GiB, TiB, PiB, EiB, ZiB and YiB) then the number is interpreted as the size of the partition in bytes and it is then aligned according to the device I/O limits. A '+' can be used instead of a number to enlarge the partition as much as possible. Note '+' is equivalent to the default

behaviour for a new partition; existing partitions will be resized as required.

The partition *type* is given in hex for MBR (DOS) where 0x prefix is optional; a GUID string for GPT; a shortcut or an alias. It's recommended to use two letters for MBR hex codes to avoid collision between deprecated shortcut 'E' and '0E' MBR hex code. For backward compatibility **sfdisk** tries to interpret *type* as a shortcut as a first possibility in partitioning scripts although on other places (e.g. **--part-type** command) it tries shortcuts as the last possibility.

Since v2.36 libfdisk supports partition type aliases as extension to shortcuts. The alias is a simple human readable word (e.g. "linux").

Since v2.37 libfdisk supports partition type names on input, ignoring the case of the characters and all non-alphanumeric and non-digit characters in the name (e.g. "Linux /usr x86" is the same as "linux usr-x86").

Supported shortcuts and aliases:

L – alias 'linux'

Linux; means 83 for MBR and 0FC63DAF-8483-4772-8E79-3D69D8477DE4 for GPT.

S – alias 'swap'

swap area; means 82 for MBR and 0657FD6D-A4AB-43C4-84E5-0933C84B4F4F for GPT

Ex – alias 'extended'

MBR extended partition; means 05 for MBR. The original shortcut 'E' is deprecated due to collision with 0x0E MBR partition type.

H – alias 'home'

home partition; means 933AC7E1-2EB4-4F13-B844-0E14E2AEF915 for GPT

U – alias 'uefi'

EFI System partition, means EF for MBR and C12A7328-F81F-11D2-BA4B-00A0C93EC93B for GPT

R – alias 'raid'

Linux RAID; means FD for MBR and A19D880F-05FC-4D3B-A006-743F0F84911E for GPT

V – alias 'lvm'

LVM; means 8E for MBR and E6D6D379-F507-44C2-A23C-238F2A3DF928 for GPT

The default *type* value is *linux*.

The shortcut 'X' for Linux extended partition (85) is deprecated in favour of 'Ex'.

bootable is specified as [*|–], with as default not-bootable. The value of this field is irrelevant for Linux – when Linux runs it has been booted already – but it might play a role for certain boot loaders and for other operating systems.

Named-fields format

This format is more readable, robust, extensible and allows specifying additional information (e.g., a UUID). It is recommended to use this format to keep your scripts more readable.

[*device* :] *name*[=*value*], ...

The *device* field is optional. **sfdisk** extracts the partition number from the device name. It allows specifying

the partitions in random order. This functionality is mostly used by **--dump**. Don't use it if you are not sure.

The *value* can be between quotation marks (e.g., name="This is partition name"). The fields **start=** and **size=** support '+' and '-' in the same way as **Unnamed-fields format**.

The currently supported fields are:

start=number

The first non-assigned sector aligned according to device I/O limits. The default start offset for the first partition is 1 MiB. If the offset is followed by the multiplicative suffixes (KiB, MiB, GiB, TiB, PiB, EiB, ZiB and YiB), then the number is interpreted as offset in bytes.

size=number

Specify the partition size in sectors. The number may be followed by the multiplicative suffixes (KiB, MiB, GiB, TiB, PiB, EiB, ZiB and YiB), then it's interpreted as size in bytes and the size is aligned according to device I/O limits.

bootable

Mark the partition as bootable.

attrs=string

Partition attributes, usually GPT partition attribute bits. See **--part-attrs** for more details about the GPT-bits string format.

uuid=string

GPT partition UUID.

name=string

GPT partition name.

type=code

A hexadecimal number (without 0x) for an MBR partition, a GUID for a GPT partition, a shortcut as for unnamed-fields format or a type name (e.g. type="Linux /usr (x86)"). See above the section about the unnamed-fields format for more details. For backward compatibility the **Id=** field has the same meaning.

EMPTY DISK LABEL

sfdisk does not create partition table without partitions by default. The lines with partitions are expected in the script by default. The empty partition table has to be explicitly requested by "label: <name>" script header line without any partitions lines. For example:

```
echo 'label: gpt' | sfdisk /dev/sdb
```

creates empty GPT partition table. Note that the **--append** disables this feature.

BACKING UP THE PARTITION TABLE

It is recommended to save the layout of your devices. **sfdisk** supports two ways.

Dump in sfdisk compatible format

Use the **--dump** command to save a description of the device layout to a text file. The dump format is suitable for later **sfdisk** input. For example:

```
sfdisk --dump /dev/sda > sda.dump
```

This can later be restored by:

```
sfdisk /dev/sda < sda.dump
```

Full binary backup

If you want to do a full binary backup of all sectors where the partition table is stored, then use the **--backup-pt-sectors** command. It writes the sectors to `~/sfdisk-<device>-<offset>.bak` files. The default name of the backup file can be changed with the **--backup-file** option. The backup files contain only raw data from the *device*. For example:

```
sfdisk --backup-pt-sectors /dev/sda
```

The GPT header can later be restored by:

```
dd if=~/sfdisk-sda-0x00000200.bak of=/dev/sda seek=$((0x00000200)) bs=1  
conv=notrunc
```

It's also possible to use the **--backup** option to create the same backup immediately after startup for other **sfdisk** commands. For example, backup partition table before deleting all partitions from partition table:

```
sfdisk --backup --delete /dev/sda
```

The same concept of backup files is used by **wipefs(8)**.

Note that **sfdisk** since version 2.26 no longer provides the **-I** option to restore sectors. **dd(1)** provides all necessary functionality.

COLORS

The output colorization is implemented by **terminal-colors.d(5)** functionality. Implicit coloring can be disabled by an empty file

```
/etc/terminal-colors.d/sfdisk.disable
```

for the **sfdisk** command or for all tools by

```
/etc/terminal-colors.d/disable
```

The user-specific `$XDG_CONFIG_HOME/terminal-colors.d` or `$HOME/.config/terminal-colors.d` overrides the global setting.

Note that the output colorization may be enabled by default, and in this case `terminal-colors.d` directories do not have to exist yet.

The logical color names supported by **sfdisk** are:

header

The header of the output tables.

warn

The warning messages.

welcome

The welcome message.

ENVIRONMENT

SFDISK_DEBUG=all

enables **sfdisk** debug output.

LIBFDISK_DEBUG=all
enables libfdisk debug output.

LIBBLKID_DEBUG=all
enables libblkid debug output.

LIBSMARTCOLS_DEBUG=all
enables libsmartcols debug output.

LOCK_BLOCK_DEVICE=<mode>
use exclusive BSD lock. The mode is "1" or "0". See **--lock** for more details.

NOTES

Since version 2.26 **sfdisk** no longer provides the **-R** or **--re-read** option to force the kernel to reread the partition table. Use **blockdev --rereadpt** instead.

Since version 2.26 **sfdisk** does not provide the **--DOS**, **--IBM**, **--DOS-extended**, **--unhide**, **--show-extended**, **--cylinders**, **--heads**, **--sectors**, **--inside-outer**, **--not-inside-outer** options.

EXAMPLES

sfdisk --list --label-nested=mbr /dev/sda
Print protective MBR on device with GPT disk label.

echo -e ',10M,L\n,10M,L\n,+,\n' | sfdisk /dev/sdc
Create three Linux partitions, with the default start, the size of the first two partitions is 10MiB, and the last partition fills all available space on the device.

echo -e 'size=10M, type=L\n size=10M, type=L\n size=+\n' | sfdisk /dev/sdc
The same as the previous example, but in named-fields format.

echo -e 'type=swap' | sfdisk -N 3 /dev/sdc
Set type of the 3rd partition to 'swap'.

sfdisk --part-type /dev/sdc 3 swap
The same as the previous example, but without script use.

sfdisk --delete /dev/sdc 2
Delete 2nd partition.

echo "," | sfdisk -N 3 --move-data /dev/sdc
Enlarge 3rd partition in both directions, move start to use free space before the partition and enlarge the size to use all free space after to the partition, and move partition data too.

AUTHORS

Karel Zak <kzak@redhat.com>

The current **sfdisk** implementation is based on the original **sfdisk** from Andries E. Brouwer.

SEE ALSO

fdisk(8), cfdisk(8), parted(8), partprobe(8), partx(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **sfdisk** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](#) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME**sftp-server** — OpenSSH SFTP server subsystem**SYNOPSIS**

```
sftp-server [-ehR] [-d start_directory] [-f log_facility] [-l log_level]
            [-P denied_requests] [-p allowed_requests] [-u umask]
sftp-server -Q protocol_feature
```

DESCRIPTION

sftp-server is a program that speaks the server side of SFTP protocol to stdout and expects client requests from stdin. **sftp-server** is not intended to be called directly, but from sshd(8) using the **Subsystem** option.

Command-line flags to **sftp-server** should be specified in the **Subsystem** declaration. See sshd_config(5) for more information.

Valid options are:

-d start_directory

Specifies an alternate starting directory for users. The pathname may contain the following tokens that are expanded at runtime: %% is replaced by a literal '%', %d is replaced by the home directory of the user being authenticated, and %u is replaced by the username of that user. The default is to use the user's home directory. This option is useful in conjunction with the sshd_config(5) **ChrootDirectory** option.

-e Causes **sftp-server** to print logging information to stderr instead of syslog for debugging.

-f log_facility

Specifies the facility code that is used when logging messages from **sftp-server**. The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.

-h Displays **sftp-server** usage information.

-l log_level

Specifies which messages will be logged by **sftp-server**. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. INFO and VERBOSE log transactions that **sftp-server** performs on behalf of the client. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. The default is ERROR.

-P denied_requests

Specifies a comma-separated list of SFTP protocol requests that are banned by the server. **sftp-server** will reply to any denied request with a failure. The **-Q** flag can be used to determine the supported request types. If both denied and allowed lists are specified, then the denied list is applied before the allowed list.

-p allowed_requests

Specifies a comma-separated list of SFTP protocol requests that are permitted by the server. All request types that are not on the allowed list will be logged and replied to with a failure message.

Care must be taken when using this feature to ensure that requests made implicitly by SFTP clients are permitted.

-Q protocol_feature

Queries protocol features supported by **sftp-server**. At present the only feature that may be queried is "requests", which may be used to deny or allow specific requests (flags **-P** and **-p** re-

spectively).

-R Places this instance of **sftp-server** into a read-only mode. Attempts to open files for writing, as well as other operations that change the state of the filesystem, will be denied.

-u umask

Sets an explicit umask(2) to be applied to newly-created files and directories, instead of the user's default mask.

On some systems, **sftp-server** must be able to access /dev/log for logging to work, and use of **sftp-server** in a chroot configuration therefore requires that **syslogd(8)** establish a logging socket inside the chroot directory.

SEE ALSO

sftp(1), **ssh(1)**, **sshd_config(5)**, **sshd(8)**

T. Ylonen and S. Lehtinen, *SSH File Transfer Protocol*, draft-ietf-secsh-filexfer-02.txt, October 2001, work in progress material.

HISTORY

sftp-server first appeared in OpenBSD 2.8.

AUTHORS

Markus Friedl <markus@openbsd.org>

NAME

sftp — OpenSSH secure file transfer

SYNOPSIS

```
sftp [-46AaCfNpqrv] [-B buffer_size] [-b batchfile] [-c cipher]
      [-D sftp_server_path] [-F ssh_config] [-i identity_file]
      [-J destination] [-l limit] [-o ssh_option] [-P port] [-R num_requests]
      [-s program] [-s subsystem | sftp_server] destination
```

DESCRIPTION

sftp is a file transfer program, similar to **ftp(1)**, which performs all operations over an encrypted **ssh(1)** transport. It may also use many features of **ssh**, such as public key authentication and compression.

The *destination* may be specified either as [user@]host[:path] or as a URI in the form sftp://[user@]host[:port][/path].

If the *destination* includes a *path* and it is not a directory, **sftp** will retrieve files automatically if a non-interactive authentication method is used; otherwise it will do so after successful interactive authentication.

If no *path* is specified, or if the *path* is a directory, **sftp** will log in to the specified *host* and enter interactive command mode, changing to the remote directory if one was specified. An optional trailing slash can be used to force the *path* to be interpreted as a directory.

Since the destination formats use colon characters to delimit host names from path names or port numbers, IPv6 addresses must be enclosed in square brackets to avoid ambiguity.

The options are as follows:

-4 Forces **sftp** to use IPv4 addresses only.

-6 Forces **sftp** to use IPv6 addresses only.

-A Allows forwarding of **ssh-agent(1)** to the remote system. The default is not to forward an authentication agent.

-a Attempt to continue interrupted transfers rather than overwriting existing partial or complete copies of files. If the partial contents differ from those being transferred, then the resultant file is likely to be corrupt.

-B *buffer_size*

Specify the size of the buffer that **sftp** uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 32768 bytes.

-b *batchfile*

Batch mode reads a series of commands from an input *batchfile* instead of *stdin*. Since it lacks user interaction it should be used in conjunction with non-interactive authentication to obviate the need to enter a password at connection time (see **sshd(8)** and **ssh-keygen(1)** for details).

A *batchfile* of ‘-’ may be used to indicate standard input. **sftp** will abort if any of the following commands fail: **get**, **put**, **reget**, **reput**, **rename**, **ln**, **rm**, **mkdir**, **chdir**, **ls**, **lchdir**, **chmod**, **chown**, **chgrp**, **lpwd**, **df**, **symlink**, and **lmdir**.

Termination on error can be suppressed on a command by command basis by prefixing the command with a ‘-’ character (for example, **-rm /tmp/blah***). Echo of the command may be suppressed by prefixing the command with a ‘@’ character. These two prefixes may be combined in any order, for example **-@ls /bsd**.

- C** Enables compression (via ssh's **-C** flag).
- c cipher** Selects the cipher to use for encrypting the data transfers. This option is directly passed to ssh(1).
- D sftp_server_path** Connect directly to a local sftp server (rather than via ssh(1)). This option may be useful in debugging the client and server.
- F ssh_config** Specifies an alternative per-user configuration file for ssh(1). This option is directly passed to ssh(1).
- f** Requests that files be flushed to disk immediately after transfer. When uploading files, this feature is only enabled if the server implements the "fsync@openssh.com" extension.
- i identity_file** Selects the file from which the identity (private key) for public key authentication is read. This option is directly passed to ssh(1).
- J destination** Connect to the target host by first making an **sftp** connection to the jump host described by *destination* and then establishing a TCP forwarding to the ultimate destination from there. Multiple jump hops may be specified separated by comma characters. This is a shortcut to specify a **ProxyJump** configuration directive. This option is directly passed to ssh(1).
- l limit** Limits the used bandwidth, specified in Kbit/s.
- N** Disables quiet mode, e.g. to override the implicit quiet mode set by the **-b** flag.
- o ssh_option** Can be used to pass options to **ssh** in the format used in **ssh_config(5)**. This is useful for specifying options for which there is no separate **sftp** command-line flag. For example, to specify an alternate port use: **sftp -oPort=24**. For full details of the options listed below, and their possible values, see **ssh_config(5)**.
 - AddressFamily
 - BatchMode
 - BindAddress
 - BindInterface
 - CanonicalDomains
 - CanonicalFallbackLocal
 - CanonicalizeHostname
 - CanonicalizeMaxDots
 - CanonicalizePermittedCNAMES
 - CASignatureAlgorithms
 - CertificateFile
 - CheckHostIP
 - Ciphers
 - Compression
 - ConnectionAttempts
 - ConnectTimeout
 - ControlMaster

ControlPath
ControlPersist
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIDelegateCredentials
HashKnownHosts
Host
HostbasedAcceptedAlgorithms
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
Hostname
IdentitiesOnly
IdentityAgent
IdentityFile
IPQoS
KbdInteractiveAuthentication
KbdInteractiveDevices
KexAlgorithms
KnownHostsCommand
LogLevel
MACs
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
PKCS11Provider
Port
PreferredAuthentications
ProxyCommand
ProxyJump
PubkeyAcceptedAlgorithms
PubkeyAuthentication
RekeyLimit
SendEnv
ServerAliveInterval
ServerAliveCountMax
SetEnv
StrictHostKeyChecking
TCPKeepAlive
UpdateHostKeys
User
UserKnownHostsFile
VerifyHostKeyDNS

-P port

Specifies the port to connect to on the remote host.

-p Preserves modification times, access times, and modes from the original files transferred.

-q Quiet mode: disables the progress meter as well as warning and diagnostic messages from ssh(1).

-R num_requests

Specify how many requests may be outstanding at any one time. Increasing this may slightly improve file transfer speed but will increase memory usage. The default is 64 outstanding requests.

- r** Recursively copy entire directories when uploading and downloading. Note that **sftp** does not follow symbolic links encountered in the tree traversal.
- s program** Name of the *program* to use for the encrypted connection. The program must understand **ssh(1)** options.
- s subsystem | sftp_server** Specifies the SSH2 subsystem or the path for an sftp server on the remote host. A path is useful when the remote **sshd(8)** does not have an sftp subsystem configured.
- v** Raise logging level. This option is also passed to **ssh**.

INTERACTIVE COMMANDS

Once in interactive mode, **sftp** understands a set of commands similar to those of **f tp(1)**. Commands are case insensitive. Pathnames that contain spaces must be enclosed in quotes. Any special characters contained within pathnames that are recognized by **glob(3)** must be escaped with backslashes ('\'').

bye Quit **sftp**.

cd [path]

Change remote directory to *path*. If *path* is not specified, then change directory to the one the session started in.

chgrp [-h] grp path

Change group of file *path* to *grp*. *path* may contain **glob(7)** characters and may match multiple files. *grp* must be a numeric GID.

If the **-h** flag is specified, then symlinks will not be followed. Note that this is only supported by servers that implement the "lsetstat@openssh.com" extension.

chmod [-h] mode path

Change permissions of file *path* to *mode*. *path* may contain **glob(7)** characters and may match multiple files.

If the **-h** flag is specified, then symlinks will not be followed. Note that this is only supported by servers that implement the "lsetstat@openssh.com" extension.

chown [-h] own path

Change owner of file *path* to *own*. *path* may contain **glob(7)** characters and may match multiple files. *own* must be a numeric UID.

If the **-h** flag is specified, then symlinks will not be followed. Note that this is only supported by servers that implement the "lsetstat@openssh.com" extension.

df [-hi] [path]

Display usage information for the filesystem holding the current directory (or *path* if specified). If the **-h** flag is specified, the capacity information will be displayed using "human-readable" suffixes.

The **-i** flag requests display of inode information in addition to capacity information. This command is only supported on servers that implement the "statvfs@openssh.com" extension.

exit Quit **sftp**.

get [-afpR] remote-path [local-path]

Retrieve the *remote-path* and store it on the local machine. If the local path name is not specified, it is given the same name it has on the remote machine. *remote-path* may contain **glob(7)** characters and may match multiple files. If it does and *local-path* is specified, then *local-path* must specify a directory.

If the **-a** flag is specified, then attempt to resume partial transfers of existing files. Note that resumption assumes that any partial copy of the local file matches the remote copy. If the remote file contents differ from the partial local copy then the resultant file is likely to be corrupt.

If the **-f** flag is specified, then `fsync(2)` will be called after the file transfer has completed to flush the file to disk.

If the **-p** flag is specified, then full file permissions and access times are copied too.

If the **-R** flag is specified then directories will be copied recursively. Note that **sftp** does not follow symbolic links when performing recursive transfers.

help Display help text.

lcd [*path*]

Change local directory to *path*. If *path* is not specified, then change directory to the local user's home directory.

l1s [*ls-options* [*path*]]

Display local directory listing of either *path* or current directory if *path* is not specified. *ls-options* may contain any flags supported by the local system's `ls(1)` command. *path* may contain `glob(7)` characters and may match multiple files.

lmkdir *path*

Create local directory specified by *path*.

ln [**-s**] *oldpath newpath*

Create a link from *oldpath* to *newpath*. If the **-s** flag is specified the created link is a symbolic link, otherwise it is a hard link.

lpwd Print local working directory.

ls [**-1afhlnrst**] [*path*]

Display a remote directory listing of either *path* or the current directory if *path* is not specified. *path* may contain `glob(7)` characters and may match multiple files.

The following flags are recognized and alter the behaviour of **ls** accordingly:

-1 Produce single columnar output.

-a List files beginning with a dot ('.') .

-f Do not sort the listing. The default sort order is lexicographical.

-h When used with a long format option, use unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte, and Exabyte in order to reduce the number of digits to four or fewer using powers of 2 for sizes (K=1024, M=1048576, etc.).

-1 Display additional details including permissions and ownership information.

-n Produce a long listing with user and group information presented numerically.

-r Reverse the sort order of the listing.

-s Sort the listing by file size.

-t Sort the listing by last modification time.

lumask *umask*

Set local umask to *umask*.

mkdir *path*

Create remote directory specified by *path*.

progress

Toggle display of progress meter.

put [**-afpR**] *local-path* [*remote-path*]

Upload *local-path* and store it on the remote machine. If the remote path name is not specified, it is given the same name it has on the local machine. *local-path* may contain glob(7) characters and may match multiple files. If it does and *remote-path* is specified, then *remote-path* must specify a directory.

If the **-a** flag is specified, then attempt to resume partial transfers of existing files. Note that resumption assumes that any partial copy of the remote file matches the local copy. If the local file contents differ from the remote local copy then the resultant file is likely to be corrupt.

If the **-f** flag is specified, then a request will be sent to the server to call fsync(2) after the file has been transferred. Note that this is only supported by servers that implement the "fsync@openssh.com" extension.

If the **-p** flag is specified, then full file permissions and access times are copied too.

If the **-R** flag is specified then directories will be copied recursively. Note that **sftp** does not follow symbolic links when performing recursive transfers.

pwd Display remote working directory.**quit** Quit **sftp**.**reget** [**-fpR**] *remote-path* [*local-path*]

Resume download of *remote-path*. Equivalent to **get** with the **-a** flag set.

reput [**-fpR**] *local-path* [*remote-path*]

Resume upload of *local-path*. Equivalent to **put** with the **-a** flag set.

rename *oldpath newpath*

Rename remote file from *oldpath* to *newpath*.

rm *path*

Delete remote file specified by *path*.

rmdir *path*

Remove remote directory specified by *path*.

symlink *oldpath newpath*

Create a symbolic link from *oldpath* to *newpath*.

version

Display the **sftp** protocol version.

!command

Execute *command* in local shell.

!

Escape to local shell.

?

Synonym for help.

SEE ALSO

ftp(1), **ls(1)**, **scp(1)**, **ssh(1)**, **ssh-add(1)**, **ssh-keygen(1)**, **ssh_config(5)**, **glob(7)**, **sftp-server(8)**, **sshd(8)**

T. Ylonen and S. Lehtinen, *SSH File Transfer Protocol*, draft-ietf-secsh-filexfer-00.txt, January 2001, work in progress material.

NAME

dash — command interpreter (shell)

SYNOPSIS

```
dash [-aCefnuvxIimqVEbp] [+aCefnuvxIimqVEbp] [-o option_name]
      [+o option_name] [command_file [argument ...]]
dash -c[-aCefnuvxIimqVEbp] [+aCefnuvxIimqVEbp] [-o option_name]
      [+o option_name] command_string [command_name [argument ...]]
dash -s[-aCefnuvxIimqVEbp] [+aCefnuvxIimqVEbp] [-o option_name]
      [+o option_name] [argument ...]
```

DESCRIPTION

dash is the standard command interpreter for the system. The current version of **dash** is in the process of being changed to conform with the POSIX 1003.2 and 1003.2a specifications for the shell. This version has many features which make it appear similar in some respects to the Korn shell, but it is not a Korn shell clone (see **ksh(1)**). Only features designated by POSIX, plus a few Berkeley extensions, are being incorporated into this shell. This man page is not intended to be a tutorial or a complete specification of the shell.

Overview

The shell is a command that reads lines from either a file or the terminal, interprets them, and generally executes other commands. It is the program that is running when a user logs into the system (although a user can select a different shell with the **chsh(1)** command). The shell implements a language that has flow control constructs, a macro facility that provides a variety of features in addition to data storage, along with built in history and line editing capabilities. It incorporates many features to aid interactive use and has the advantage that the interpretative language is common to both interactive and non-interactive use (shell scripts). That is, commands can be typed directly to the running shell or can be put into a file and the file can be executed directly by the shell.

Invocation

If no args are present and if the standard input of the shell is connected to a terminal (or if the **-i** flag is set), and the **-c** option is not present, the shell is considered an interactive shell. An interactive shell generally prompts before each command and handles programming and command errors differently (as described below). When first starting, the shell inspects argument 0, and if it begins with a dash ‘-’, the shell is also considered a login shell. This is normally done automatically by the system when the user first logs in. A login shell first reads commands from the files **/etc/profile** and **.profile** if they exist. If the environment variable **ENV** is set on entry to an interactive shell, or is set in the **.profile** of a login shell, the shell next reads commands from the file named in **ENV**. Therefore, a user should place commands that are to be executed only at login time in the **.profile** file, and commands that are executed for every interactive shell inside the **ENV** file. To set the **ENV** variable to some file, place the following line in your **.profile** of your home directory

```
ENV=$HOME/.shinit; export ENV
```

substituting for “**.shinit**” any filename you wish.

If command line arguments besides the options have been specified, then the shell treats the first argument as the name of a file from which to read commands (a shell script), and the remaining arguments are set as the positional parameters of the shell (**\$1**, **\$2**, etc). Otherwise, the shell reads commands from its standard input.

Argument List Processing

All of the single letter options that have a corresponding name can be used as an argument to the **-o** option. The set **-o** name is provided next to the single letter option in the description below. Specifying a dash “-” turns the option on, while using a plus “+” disables the option. The following options can be set from the

command line or with the **set** builtin (described later).

-a <i>alllexport</i>	Export all variables assigned to.
-c	Read commands from the <i>command_string</i> operand instead of from the standard input. Special parameter 0 will be set from the <i>command_name</i> operand and the positional parameters (\$1, \$2, etc.) set from the remaining argument operands.
-C <i>noclobber</i>	Don't overwrite existing files with ">".
-e <i>errexit</i>	If not interactive, exit immediately if any untested command fails. The exit status of a command is considered to be explicitly tested if the command is used to control an if , elif , while , or until ; or if the command is the left hand operand of an “&&” or “ ” operator.
-f <i>noglob</i>	Disable pathname expansion.
-n <i>noexec</i>	If not interactive, read commands but do not execute them. This is useful for checking the syntax of shell scripts.
-u <i>nounset</i>	Write a message to standard error when attempting to expand a variable that is not set, and if the shell is not interactive, exit immediately.
-v <i>verbose</i>	The shell writes its input to standard error as it is read. Useful for debugging.
-x <i>xtrace</i>	Write each command to standard error (preceded by a ‘+’) before it is executed. Useful for debugging.
-I <i>ignoreeof</i>	Ignore EOF's from input when interactive.
-i <i>interactive</i>	Force the shell to behave interactively.
-l	Make dash act as if it had been invoked as a login shell.
-m <i>monitor</i>	Turn on job control (set automatically when interactive).
-s <i>stdin</i>	Read commands from standard input (set automatically if no file arguments are present). This option has no effect when the shell has already started running (i.e. with set).
-v <i>vi</i>	Enable the built-in vi (1) command line editor (disables -E if it has been set).
-E <i>emacs</i>	Enable the built-in emacs (1) command line editor (disables -v if it has been set).
-b <i>notify</i>	Enable asynchronous notification of background job completion. (UNIMPLEMENTED for 4.4alpha)
-p <i>privileged</i>	Do not attempt to reset effective uid if it does not match uid. This is not set by default to help avoid incorrect usage by setuid root programs via system (3) or popen (3).

Lexical Structure

The shell reads input in terms of lines from a file and breaks it up into words at whitespace (blanks and tabs), and at certain sequences of characters that are special to the shell called “operators”. There are two types of operators: control operators and redirection operators (their meaning is discussed later). Following is a list of operators:

Control operators:

```
& && ( ) ; ; | || <newline>
```

Redirection operators:

```
< > >| << >> <& >& <<- >>
```

Quoting

Quoting is used to remove the special meaning of certain characters or words to the shell, such as operators, whitespace, or keywords. There are three types of quoting: matched single quotes, matched double quotes, and backslash.

Backslash

A backslash preserves the literal meaning of the following character, with the exception of `<newline>`. A backslash preceding a `<newline>` is treated as a line continuation.

Single Quotes

Enclosing characters in single quotes preserves the literal meaning of all the characters (except single quotes, making it impossible to put single-quotes in a single-quoted string).

Double Quotes

Enclosing characters within double quotes preserves the literal meaning of all characters except dollarsign (\$), backquote (`), and backslash (\). The backslash inside double quotes is historically weird, and serves to quote only the following characters:

```
$ ` " \ <newline>
```

Otherwise it remains literal.

Reserved Words

Reserved words are words that have special meaning to the shell and are recognized at the beginning of a line and after a control operator. The following are reserved words:

```
!      elif    fi      while  case
else   for     then    {       }
do     done    until   if     esac
```

Their meaning is discussed later.

Aliases

An alias is a name and corresponding value set using the `alias(1)` builtin command. Whenever a reserved word may occur (see above), and after checking for reserved words, the shell checks the word to see if it matches an alias. If it does, it replaces it in the input stream with its value. For example, if there is an alias called “`lf`” with the value “`ls -F`”, then the input:

```
lf foobar <return>
```

would become

```
ls -F foobar <return>
```

Aliases provide a convenient way for naive users to create shorthands for commands without having to learn how to create functions with arguments. They can also be used to create lexically obscure code. This use is discouraged.

Commands

The shell interprets the words it reads according to a language, the specification of which is outside the scope of this man page (refer to the BNF in the POSIX 1003.2 document). Essentially though, a line is read and if

the first word of the line (or after a control operator) is not a reserved word, then the shell has recognized a simple command. Otherwise, a complex command or some other special construct may have been recognized.

Simple Commands

If a simple command has been recognized, the shell performs the following actions:

1. Leading words of the form “name=value” are stripped off and assigned to the environment of the simple command. Redirection operators and their arguments (as described below) are stripped off and saved for processing.
2. The remaining words are expanded as described in the section called “Expansions”, and the first remaining word is considered the command name and the command is located. The remaining words are considered the arguments of the command. If no command name resulted, then the “name=value” variable assignments recognized in item 1 affect the current shell.
3. Redirections are performed as described in the next section.

Redirections

Redirections are used to change where a command reads its input or sends its output. In general, redirections open, close, or duplicate an existing reference to a file. The overall format used for redirection is:

`[n] redir-op file`

where *redir-op* is one of the redirection operators mentioned previously. Following is a list of the possible redirections. The [n] is an optional number between 0 and 9, as in ‘3’ (not ‘[3]’), that refers to a file descriptor.

<code>[n]> file</code>	Redirect standard output (or n) to file.
<code>[n]> file</code>	Same, but override the -C option.
<code>[n]>> file</code>	Append standard output (or n) to file.
<code>[n]< file</code>	Redirect standard input (or n) from file.
<code>[n]<&n2</code>	Copy file descriptor n2 as stdout (or fd n1). fd n2.
<code>[n]<&-</code>	Close standard input (or n).
<code>[n]>&n2</code>	Copy file descriptor n2 as stdin (or fd n1). fd n2.
<code>[n]>&-</code>	Close standard output (or n).
<code>[n]<> file</code>	Open file for reading and writing on standard input (or n).

The following redirection is often called a “here-document”.

```
[n]<< delimiter
    here-doc-text ...
delimiter
```

All the text on successive lines up to the delimiter is saved away and made available to the command on standard input, or file descriptor n if it is specified. If the delimiter as specified on the initial line is quoted, then the here-doc-text is treated literally, otherwise the text is subjected to parameter expansion, command substitution, and arithmetic expansion (as described in the section on “Expansions”). If the operator is “<<-” instead of “<<”, then leading tabs in the here-doc-text are stripped.

Search and Execution

There are three types of commands: shell functions, builtin commands, and normal programs – and the command is searched for (by name) in that order. They each are executed in a different way.

When a shell function is executed, all of the shell positional parameters (except \$0, which remains unchanged) are set to the arguments of the shell function. The variables which are explicitly placed in the environment of the command (by placing assignments to them before the function name) are made local to the function and are set to the values given. Then the command given in the function definition is executed. The positional parameters are restored to their original values when the command completes. This all occurs within the current shell.

Shell builtins are executed internally to the shell, without spawning a new process.

Otherwise, if the command name doesn't match a function or builtin, the command is searched for as a normal program in the file system (as described in the next section). When a normal program is executed, the shell runs the program, passing the arguments and the environment to the program. If the program is not a normal executable file (i.e., if it does not begin with the "magic number" whose ASCII representation is "#!", so `execve(2)` returns `ENOEXEC` then) the shell will interpret the program in a subshell. The child shell will reinitialize itself in this case, so that the effect will be as if a new shell had been invoked to handle the ad-hoc shell script, except that the location of hashed commands located in the parent shell will be remembered by the child.

Note that previous versions of this document and the source code itself misleadingly and sporadically refer to a shell script without a magic number as a "shell procedure".

Path Search

When locating a command, the shell first looks to see if it has a shell function by that name. Then it looks for a builtin command by that name. If a builtin command is not found, one of two things happen:

1. Command names containing a slash are simply executed without performing any searches.
2. The shell searches each entry in `PATH` in turn for the command. The value of the `PATH` variable should be a series of entries separated by colons. Each entry consists of a directory name. The current directory may be indicated implicitly by an empty directory name, or explicitly by a single period.

Command Exit Status

Each command has an exit status that can influence the behaviour of other shell commands. The paradigm is that a command exits with zero for normal or success, and non-zero for failure, error, or a false indication. The man page for each command should indicate the various exit codes and what they mean. Additionally, the builtin commands return exit codes, as does an executed shell function.

If a command consists entirely of variable assignments then the exit status of the command is that of the last command substitution if any, otherwise 0.

Complex Commands

Complex commands are combinations of simple commands with control operators or reserved words, together creating a larger complex command. More generally, a command is one of the following:

- simple command
- pipeline
- list or compound-list
- compound command

- function definition

Unless otherwise stated, the exit status of a command is that of the last simple command executed by the command.

Pipelines

A pipeline is a sequence of one or more commands separated by the control operator `|`. The standard output of all but the last command is connected to the standard input of the next command. The standard output of the last command is inherited from the shell, as usual.

The format for a pipeline is:

```
[ ! ] command1 [| command2 ... ]
```

The standard output of `command1` is connected to the standard input of `command2`. The standard input, standard output, or both of a command is considered to be assigned by the pipeline before any redirection specified by redirection operators that are part of the command.

If the pipeline is not in the background (discussed later), the shell waits for all commands to complete.

If the reserved word `!` does not precede the pipeline, the exit status is the exit status of the last command specified in the pipeline. Otherwise, the exit status is the logical NOT of the exit status of the last command. That is, if the last command returns zero, the exit status is 1; if the last command returns greater than zero, the exit status is zero.

Because pipeline assignment of standard input or standard output or both takes place before redirection, it can be modified by redirection. For example:

```
$ command1 2>&1 | command2
```

sends both the standard output and standard error of `command1` to the standard input of `command2`.

`A ;` or `<newline>` terminator causes the preceding AND-OR-list (described next) to be executed sequentially; a `&` causes asynchronous execution of the preceding AND-OR-list.

Note that unlike some other shells, each process in the pipeline is a child of the invoking shell (unless it is a shell builtin, in which case it executes in the current shell – but any effect it has on the environment is wiped).

Background Commands – &

If a command is terminated by the control operator ampersand (`&`), the shell executes the command asynchronously – that is, the shell does not wait for the command to finish before executing the next command.

The format for running a command in background is:

```
command1 & [ command2 & ... ]
```

If the shell is not interactive, the standard input of an asynchronous command is set to `/dev/null`.

Lists – Generally Speaking

A list is a sequence of zero or more commands separated by newlines, semicolons, or ampersands, and optionally terminated by one of these three characters. The commands in a list are executed in the order they are written. If command is followed by an ampersand, the shell starts the command and immediately proceeds onto the next command; otherwise it waits for the command to terminate before proceeding to the next one.

Short-Circuit List Operators

“`&&`” and “`||`” are AND-OR list operators. “`&&`” executes the first command, and then executes the second command if and only if the exit status of the first command is zero. “`||`” is similar, but executes the second command if and only if the exit status of the first command is nonzero. “`&&`” and “`||`” both have the same priority.

Flow-Control Constructs – if, while, for, case

The syntax of the if command is

```
if list
then list
[ elif list
then    list ] ...
[ else list ]
fi
```

The syntax of the while command is

```
while list
do    list
done
```

The two lists are executed repeatedly while the exit status of the first list is zero. The until command is similar, but has the word until in place of while, which causes it to repeat until the exit status of the first list is zero.

The syntax of the for command is

```
for variable [ in [ word ... ] ]
do    list
done
```

The words following in are expanded, and then the list is executed repeatedly with the variable set to each word in turn. Omitting in word ... is equivalent to in “`$@`”.

The syntax of the break and continue command is

```
break [ num ]
continue [ num ]
```

Break terminates the num innermost for or while loops. Continue continues with the next iteration of the innermost loop. These are implemented as builtin commands.

The syntax of the case command is

```
case word in
[ ( ]pattern) list ;;
...
esac
```

The pattern can actually be one or more patterns (see **Shell Patterns** described later), separated by “`|`” characters. The “`(`” character before the pattern is optional.

Grouping Commands Together

Commands may be grouped by writing either

```
(list)
```

or

```
{ list; }
```

The first of these executes the commands in a subshell. Builtin commands grouped into a (list) will not affect the current shell. The second form does not fork another shell so is slightly more efficient. Grouping commands together this way allows you to redirect their output as though they were one program:

```
{ printf " hello " ; printf " world\n" ; } > greeting
```

Note that “}” must follow a control operator (here, “;”) so that it is recognized as a reserved word and not as another command argument.

Functions

The syntax of a function definition is

```
name () command
```

A function definition is an executable statement; when executed it installs a function named name and returns an exit status of zero. The command is normally a list enclosed between “{” and “}”.

Variables may be declared to be local to a function by using a local command. This should appear as the first statement of a function, and the syntax is

```
local [variable | -] . . .
```

Local is implemented as a builtin command.

When a variable is made local, it inherits the initial value and exported and readonly flags from the variable with the same name in the surrounding scope, if there is one. Otherwise, the variable is initially unset. The shell uses dynamic scoping, so that if you make the variable x local to function f, which then calls function g, references to the variable x made inside g will refer to the variable x declared inside f, not to the global variable named x.

The only special parameter that can be made local is “-”. Making “-” local any shell options that are changed via the set command inside the function to be restored to their original values when the function returns.

The syntax of the return command is

```
return [exitstatus]
```

It terminates the currently executing function. Return is implemented as a builtin command.

Variables and Parameters

The shell maintains a set of parameters. A parameter denoted by a name is called a variable. When starting up, the shell turns all the environment variables into shell variables. New variables can be set using the form

```
name=value
```

Variables set by the user must have a name consisting solely of alphabets, numerics, and underscores - the first of which must not be numeric. A parameter can also be denoted by a number or a special character as explained below.

Positional Parameters

A positional parameter is a parameter denoted by a number ($n > 0$). The shell sets these initially to the values of its command line arguments that follow the name of the shell script. The **set** builtin can also be used to set or reset them.

Special Parameters

A special parameter is a parameter denoted by one of the following special characters. The value of the parameter is listed next to its character.

*	Expands to the positional parameters, starting from one. When the expansion occurs within a double-quoted string it expands to a single field with the value of each parameter separated by the first character of the <code>IFS</code> variable, or by a <code>(space)</code> if <code>IFS</code> is unset.
@	Expands to the positional parameters, starting from one. When the expansion occurs within double-quotes, each positional parameter expands as a separate argument. If there are no positional parameters, the expansion of @ generates zero arguments, even when @ is double-quoted. What this basically means, for example, is if \$1 is "abc" and \$2 is "def ghi", then "\$@" expands to the two arguments:
	"abc" "def ghi"
#	Expands to the number of positional parameters.
?	Expands to the exit status of the most recent pipeline.
- (Hyphen.)	Expands to the current option flags (the single-letter option names concatenated into a string) as specified on invocation, by the set builtin command, or implicitly by the shell.
\$	Expands to the process ID of the invoked shell. A subshell retains the same value of \$ as its parent.
!	Expands to the process ID of the most recent background command executed from the current shell. For a pipeline, the process ID is that of the last command in the pipeline.
0 (Zero.)	Expands to the name of the shell or shell script.

Word Expansions

This clause describes the various expansions that are performed on words. Not all expansions are performed on every word, as explained later.

Tilde expansions, parameter expansions, command substitutions, arithmetic expansions, and quote removals that occur within a single word expand to a single field. It is only field splitting or pathname expansion that can create multiple fields from a single word. The single exception to this rule is the expansion of the special parameter @ within double-quotes, as was described above.

The order of word expansion is:

1. Tilde Expansion, Parameter Expansion, Command Substitution, Arithmetic Expansion (these all occur at the same time).
2. Field Splitting is performed on fields generated by step (1) unless the `IFS` variable is null.
3. Pathname Expansion (unless set `-f` is in effect).
4. Quote Removal.

The \$ character is used to introduce parameter expansion, command substitution, or arithmetic evaluation.

Tilde Expansion (substituting a user's home directory)

A word beginning with an unquoted tilde character (~) is subjected to tilde expansion. All the characters up to a slash (/) or the end of the word are treated as a username and are replaced with the user's home directory. If the username is missing (as in ~ /foobar), the tilde is replaced with the value of the `HOME` variable (the current user's home directory).

Parameter Expansion

The format for parameter expansion is as follows:

```
$ {expression}
```

where expression consists of all characters until the matching “}”. Any “}” escaped by a backslash or within a quoted string, and characters in embedded arithmetic expansions, command substitutions, and variable expansions, are not examined in determining the matching “}”.

The simplest form for parameter expansion is:

```
$ {parameter}
```

The value, if any, of parameter is substituted.

The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when parameter is followed by a character that could be interpreted as part of the name. If a parameter expansion occurs inside double-quotes:

1. Pathname expansion is not performed on the results of the expansion.
2. Field splitting is not performed on the results of the expansion, with the exception of @.

In addition, a parameter expansion can be modified by using one of the following formats.

<code>\${parameter:-word}</code>	Use Default Values. If parameter is unset or null, the expansion of word is substituted; otherwise, the value of parameter is substituted.
<code>\${parameter:=word}</code>	Assign Default Values. If parameter is unset or null, the expansion of word is assigned to parameter. In all cases, the final value of parameter is substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.
<code>\${parameter:?word}</code>	Indicate Error if Null or Unset. If parameter is unset or null, the expansion of word (or a message indicating it is unset if word is omitted) is written to standard error and the shell exits with a nonzero exit status. Otherwise, the value of parameter is substituted. An interactive shell need not exit.
<code>\${parameter:+word}</code>	Use Alternative Value. If parameter is unset or null, null is substituted; otherwise, the expansion of word is substituted.

In the parameter expansions shown previously, use of the colon in the format results in a test for a parameter that is unset or null; omission of the colon results in a test for a parameter that is only unset.

<code>\$#parameter</code>	String Length. The length in characters of the value of parameter.
---------------------------	--

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see **Shell Patterns**), rather than regular expression notation, is used to evaluate the patterns. If parameter is* or @, the result of the expansion is unspecified. Enclosing the full parameter expansion string in double-quotes does not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces has this effect.

<code>\$parameter%word</code>	Remove Smallest Suffix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the smallest portion of the suffix matched by the pattern deleted.
<code>\$parameter%%word</code>	Remove Largest Suffix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the largest portion of the suffix matched by the pattern deleted.

<code>\$(parameter#word}</code>	Remove Smallest Prefix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the smallest portion of the prefix matched by the pattern deleted.
<code>\$(parameter##word}</code>	Remove Largest Prefix Pattern. The word is expanded to produce a pattern. The parameter expansion then results in parameter, with the largest portion of the prefix matched by the pattern deleted.

Command Substitution

Command substitution allows the output of a command to be substituted in place of the command name itself. Command substitution occurs when the command is enclosed as follows:

```
$ ( command )
```

or (“backquoted” version):

```
' command '
```

The shell expands the command substitution by executing command in a subshell environment and replacing the command substitution with the standard output of the command, removing sequences of one or more `\n`s at the end of the substitution. (Embedded `\n`s before the end of the output are not removed; however, during field splitting, they may be translated into s, depending on the value of `IFS` and quoting that is in effect.)

Arithmetic Expansion

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion is as follows:

```
$ ( ( expression ) )
```

The expression is treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell expands all tokens in the expression for parameter expansion, command substitution, and quote removal.

Next, the shell treats this as an arithmetic expression and substitutes the value of the expression.

White Space Splitting (Field Splitting)

After parameter expansion, command substitution, and arithmetic expansion the shell scans the results of expansions and substitutions that did not occur in double-quotes for field splitting and multiple fields can result.

The shell treats each character of the `IFS` as a delimiter and uses the delimiters to split the results of parameter expansion and command substitution into fields.

Pathname Expansion (File Name Generation)

Unless the `-f` flag is set, file name generation is performed after word splitting is complete. Each word is viewed as a series of patterns, separated by slashes. The process of expansion replaces the word with the names of all existing files whose names can be formed by replacing each pattern with a string that matches the specified pattern. There are two restrictions on this: first, a pattern cannot match a string containing a slash, and second, a pattern cannot match a string starting with a period unless the first character of the pattern is a period. The next section describes the patterns used for both Pathname Expansion and the `case` command.

Shell Patterns

A pattern consists of normal characters, which match themselves, and meta-characters. The meta-characters are “!”, “*”, “?”, and “[”. These characters lose their special meanings if they are quoted. When command or variable substitution is performed and the dollar sign or back quotes are not double quoted, the value of

the variable or the output of the command is scanned for these characters and they are turned into meta-characters.

An asterisk (“*”) matches any string of characters. A question mark matches any single character. A left bracket (“[”) introduces a character class. The end of the character class is indicated by a (“]”); if the “]” is missing then the “[” matches a “[” rather than introducing a character class. A character class matches any of the characters between the square brackets. A range of characters may be specified using a minus sign. The character class may be complemented by making an exclamation point the first character of the character class.

To include a “[” in a character class, make it the first character listed (after the “!”, if any). To include a minus sign, make it the first or last character listed.

Builtins

This section lists the builtin commands which are builtin because they need to perform some operation that can't be performed by a separate process. In addition to these, there are several other commands that may be builtin for efficiency (e.g. `printf(1)`, `echo(1)`, `test(1)`, etc).

:

`true` A null command that returns a 0 (true) exit value.

`. file` The commands in the specified file are read and executed by the shell.

`alias [name[=string ...]]`

If `name=string` is specified, the shell defines the alias `name` with value `string`. If `justname` is specified, the value of the alias `name` is printed. With no arguments, the `alias` builtin prints the names and values of all defined aliases (see `unalias`).

`bg [job] ...`

Continue the specified jobs (or the current job if no jobs are given) in the background.

`command [-p] [-v] [-V] command [arg ...]`

Execute the specified command but ignore shell functions when searching for it. (This is useful when you have a shell function with the same name as a builtin command.)

`-p` search for command using a PATH that guarantees to find all the standard utilities.

`-v` Do not execute the command but search for the command and print the resolution of the command search. This is the same as the type builtin.

`-V` Do not execute the command but search for the command and print the absolute pathname of utilities, the name for builtins or the expansion of aliases.

`cd -`

`cd [-LP] [directory]`

Switch to the specified directory (default HOME). If an entry for CDPATH appears in the environment of the `cd` command or the shell variable CDPATH is set and the directory name does not begin with a slash, then the directories listed in CDPATH will be searched for the specified directory. The format of CDPATH is the same as that of PATH. If a single dash is specified as the argument, it will be replaced by the value of OLDPWD. The `cd` command will print out the name of the directory that it actually switched to if this is different from the name that the user gave. These may be different either because the CDPATH mechanism was used or because the argument is a single dash. The `-P` option causes the physical directory structure to be used, that is, all symbolic links are resolved to their respective values. The `-L` option turns off the effect of any preceding `-P` options.

echo [**-n**] *args...*

Print the arguments on the standard output, separated by spaces. Unless the **-n** option is present, a newline is output following the arguments.

If any of the following sequences of characters is encountered during output, the sequence is not output. Instead, the specified action is performed:

- \b A backspace character is output.
- \c Subsequent output is suppressed. This is normally used at the end of the last argument to suppress the trailing newline that **echo** would otherwise output.
- \e Outputs an escape character (ESC).
- \f Output a form feed.
- \n Output a newline character.
- \r Output a carriage return.
- \t Output a (horizontal) tab character.
- \v Output a vertical tab.

\0*digits*

Output the character whose value is given by zero to three octal digits. If there are zero digits, a nul character is output.

- \\\ Output a backslash.

All other backslash sequences elicit undefined behaviour.

eval *string* . . .

Concatenate all the arguments with spaces. Then re-parse and execute the command.

exec [*command arg* . . .]

Unless *command* is omitted, the shell process is replaced with the specified program (which must be a real program, not a shell builtin or function). Any redirections on the **exec** command are marked as permanent, so that they are not undone when the **exec** command finishes.

exit [*exitstatus*]

Terminate the shell process. If *exitstatus* is given it is used as the exit status of the shell; otherwise the exit status of the preceding command is used.

export *name* . . .**export -p**

The specified names are exported so that they will appear in the environment of subsequent commands. The only way to un-export a variable is to unset it. The shell allows the value of a variable to be set at the same time it is exported by writing

```
export name=value
```

With no arguments the export command lists the names of all exported variables. With the **-p** option specified the output will be formatted suitably for non-interactive use.

fc [**-e** *editor*] [*first [last]*]**fc -l** [**-nr**] [*first [last]*]**fc -s** [*old=new*] [*first*]

The **fc** builtin lists, or edits and re-executes, commands previously entered to an interactive shell.

-e editor

Use the editor named by editor to edit the commands. The editor string is a command name, subject to search via the PATH variable. The value in the FCEDIT variable is used as a default when **-e** is not specified. If FCEDIT is null or unset, the value of the EDITOR variable is used. If EDITOR is null or unset, ed(1) is used as the editor.

-l (ell)

List the commands rather than invoking an editor on them. The commands are written in the sequence indicated by the first and last operands, as affected by **-r**, with each command preceded by the command number.

-n Suppress command numbers when listing with **-l**.**-r** Reverse the order of the commands listed (with **-l**) or edited (with neither **-l** nor **-s**).**-s** Re-execute the command without invoking an editor.

first

last Select the commands to list or edit. The number of previous commands that can be accessed are determined by the value of the HISTSIZE variable. The value of first or last or both are one of the following:

[+]number

A positive number representing a command number; command numbers can be displayed with the **-l** option.

-number

A negative decimal number representing the command that was executed number of commands previously. For example, **-1** is the immediately previous command.

string A string indicating the most recently entered command that begins with that string. If the old=new operand is not also specified with **-s**, the string form of the first operand cannot contain an embedded equal sign.

The following environment variables affect the execution of fc:

FCEDIT Name of the editor to use.

HISTSIZE The number of previous commands that are accessible.

fg [job]

Move the specified job or the current job to the foreground.

getopts optstring var

The POSIX **getopts** command, not to be confused with the *Bell Labs* -derived getopt(1).

The first argument should be a series of letters, each of which may be optionally followed by a colon to indicate that the option requires an argument. The variable specified is set to the parsed option.

The **getopts** command deprecates the older getopt(1) utility due to its handling of arguments containing whitespace.

The **getopts** builtin may be used to obtain options and their arguments from a list of parameters. When invoked, **getopts** places the value of the next option from the option string in the list in the shell variable specified by *var* and its index in the shell variable OPTIND. When the shell is invoked, OPTIND is initialized to 1. For each option that requires an argument, the **getopts** builtin will place it in the shell variable OPTARG. If an option is not allowed for in the *optstring*, then OPTARG will be unset.

optstring is a string of recognized option letters (see `getopt(3)`). If a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space. If an option character is not found where expected, `getopts` will set the variable *var* to a “?”; `getopts` will then unset OPTARG and write output to standard error. By specifying a colon as the first character of *optstring* all errors will be ignored.

After the last option `getopts` will return a non-zero value and set *var* to “?”.

The following code fragment shows how one might process the arguments for a command that can take the options [a] and [b], and the option [c], which requires an argument.

```
while getopts abc: f
do
    case $f in
        a | b) flag=$f;;
        c)      carg=$OPTARG;;
        \?)     echo $USAGE; exit 1;;
    esac
done
shift `expr $OPTIND - 1`
```

This code will accept any of the following as equivalent:

```
cmd -acarg file file
cmd -a -c arg file file
cmd -carg -a file file
cmd -a -carg -- file file
```

`hash -rv command . . .`

The shell maintains a hash table which remembers the locations of commands. With no arguments whatsoever, the `hash` command prints out the contents of this table. Entries which have not been looked at since the last `cd` command are marked with an asterisk; it is possible for these entries to be invalid.

With arguments, the `hash` command removes the specified commands from the hash table (unless they are functions) and then locates them. With the `-v` option, `hash` prints the locations of the commands as it finds them. The `-r` option causes the `hash` command to delete all the entries in the hash table except for functions.

`pwd [-LP]`

builtin command remembers what the current directory is rather than recomputing it each time. This makes it faster. However, if the current directory is renamed, the builtin version of `pwd` will continue to print the old name for the directory. The `-P` option causes the physical value of the current working directory to be shown, that is, all symbolic links are resolved to their respective values. The `-L` option turns off the effect of any preceding `-P` options.

`read [-p prompt] [-r] variable [. . .]`

The prompt is printed if the `-p` option is specified and the standard input is a terminal. Then a line is read from the standard input. The trailing newline is deleted from the line and the line is split as described in the section on word splitting above, and the pieces are assigned to the variables in order. At least one variable must be specified. If there are more pieces than variables, the remaining pieces (along with the characters in IFS that separated them) are assigned to the last variable. If there are more variables than pieces, the remaining variables are assigned the null string. The `read` builtin will indicate success unless EOF is encountered on input, in which case failure is returned.

By default, unless the **-r** option is specified, the backslash “\” acts as an escape character, causing the following character to be treated literally. If a backslash is followed by a newline, the backslash and the newline will be deleted.

`readonly name . . .`

`readonly -p`

The specified names are marked as read only, so that they cannot be subsequently modified or unset. The shell allows the value of a variable to be set at the same time it is marked read only by writing

`readonly name=value`

With no arguments the `readonly` command lists the names of all read only variables. With the **-p** option specified the output will be formatted suitably for non-interactive use.

`printf format [arguments . . .]`

`printf` formats and prints its arguments, after the first, under control of the *format*. The *format* is a character string which contains three types of objects: plain characters, which are simply copied to standard output, character escape sequences which are converted and copied to the standard output, and format specifications, each of which causes printing of the next successive *argument*.

The *arguments* after the first are treated as strings if the corresponding format is either **b**, **c** or **s**; otherwise it is evaluated as a C constant, with the following extensions:

- A leading plus or minus sign is allowed.
- If the leading character is a single or double quote, the value is the ASCII code of the next character.

The format string is reused as often as necessary to satisfy the *arguments*. Any extra format specifications are evaluated with zero or the null string.

Character escape sequences are in backslash notation as defined in ANSI X3.159-1989 (“ANSI C89”). The characters and their meanings are as follows:

<code>\a</code>	Write a <bell> character.
<code>\b</code>	Write a <backspace> character.
<code>\e</code>	Write an <escape> (ESC) character.
<code>\f</code>	Write a <form-feed> character.
<code>\n</code>	Write a <new-line> character.
<code>\r</code>	Write a <carriage return> character.
<code>\t</code>	Write a <tab> character.
<code>\v</code>	Write a <vertical tab> character.
<code>\\\</code>	Write a backslash character.
<code>\num</code>	Write an 8-bit character whose ASCII value is the 1-, 2-, or 3-digit octal number <i>num</i> .

Each format specification is introduced by the percent character (“%”). The remainder of the format specification includes, in the following order:

Zero or more of the following flags:

- # A '#' character specifying that the value should be printed in an “alternative form”. For **b**, **c**, **d**, and **s** formats, this option has no effect. For the **o** format the precision of the number is increased to force the first character of the output string to a zero.

For the **x** (**X**) format, a non-zero result has the string 0x (0X) prepended to it. For **e**, **E**, **f**, **g**, and **G** formats, the result will always contain a decimal point, even if no digits follow the point (normally, a decimal point only appears in the results of those formats if a digit follows the decimal point). For **g** and **G** formats, trailing zeros are not removed from the result as they would otherwise be.

- A minus sign ‘-’ which specifies *left adjustment* of the output in the indicated field;
- + A ‘+’ character specifying that there should always be a sign placed before the number when using signed formats.
- ‘ ’ A space specifying that a blank should be left before a positive number for a signed format. A ‘+’ overrides a space if both are used;
- 0 A zero ‘0’ character indicating that zero-padding should be used rather than blank-padding. A ‘-’ overrides a ‘0’ if both are used;

Field Width:

An optional digit string specifying a *field width*; if the output string has fewer characters than the field width it will be blank-padded on the left (or right, if the left-adjustment indicator has been given) to make up the field width (note that a leading zero is a flag, but an embedded zero is part of a field width);

Precision:

An optional period, ‘.’, followed by an optional digit string giving a *precision* which specifies the number of digits to appear after the decimal point, for **e** and **f** formats, or the maximum number of bytes to be printed from a string (**b** and **s** formats); if the digit string is missing, the precision is treated as zero;

Format:

A character which indicates the type of format to use (one of **d****i****o****u****x****X****f****w****E****g****G****b****c****s**).

A field width or precision may be ‘*’ instead of a digit string. In this case an argument supplies the field width or precision.

The format characters and their meanings are:

- d****i****o****u****x****x** The argument is printed as a signed decimal (d or i), unsigned octal, unsigned decimal, or unsigned hexadecimal (X or x), respectively.
 - f** The argument is printed in the style [-]ddd.ddd where the number of d's after the decimal point is equal to the precision specification for the argument. If the precision is missing, 6 digits are given; if the precision is explicitly 0, no digits and no decimal point are printed.
 - e****E** The argument is printed in the style [-]d.ddde±dd where there is one digit before the decimal point and the number after is equal to the precision specification for the argument; when the precision is missing, 6 digits are produced. An upper-case E is used for an ‘E’ format.
 - g****G** The argument is printed in style **f** or in style **e** (**E**) whichever gives full precision in minimum space.
 - b** Characters from the string argument are printed with backslash-escape sequences expanded.
- The following additional backslash-escape sequences are supported:

- \c Causes **dash** to ignore any remaining characters in the string operand containing it, any remaining string operands, and any additional characters in the format operand.
- \0*num* Write an 8-bit character whose ASCII value is the 1-, 2-, or 3-digit octal number *num*.
- c The first character of *argument* is printed.
- s Characters from the string *argument* are printed until the end is reached or until the number of bytes indicated by the precision specification is reached; if the precision is omitted, all characters in the string are printed.
- % Print a ‘%’; no argument is used.

In no case does a non-existent or small field width cause truncation of a field; padding takes place only if the specified field width exceeds the actual width.

set [{ -options | +options | -- }] *arg* . . .

The **set** command performs three different functions.

With no arguments, it lists the values of all shell variables.

If options are given, it sets the specified option flags, or clears them as described in the section called **Argument List Processing**. As a special case, if the option is -o or +o and no argument is supplied, the shell prints the settings of all its options. If the option is -o, the settings are printed in a human-readable format; if the option is +o, the settings are printed in a format suitable for reinput to the shell to affect the same option settings.

The third use of the set command is to set the values of the shell’s positional parameters to the specified args. To change the positional parameters without changing any options, use “--” as the first argument to set. If no args are present, the set command will clear all the positional parameters (equivalent to executing “shift \$#”).

shift [*n*]

Shift the positional parameters *n* times. A **shift** sets the value of \$1 to the value of \$2, the value of \$2 to the value of \$3, and so on, decreasing the value of \$# by one. If *n* is greater than the number of positional parameters, **shift** will issue an error message, and exit with return status 2.

test *expression*

[*expression*]

The **test** utility evaluates the expression and, if it evaluates to true, returns a zero (true) exit status; otherwise it returns 1 (false). If there is no expression, test also returns 1 (false).

All operators and flags are separate arguments to the **test** utility.

The following primaries are used to construct expression:

- b *file* True if *file* exists and is a block special file.
- c *file* True if *file* exists and is a character special file.
- d *file* True if *file* exists and is a directory.
- e *file* True if *file* exists (regardless of type).
- f *file* True if *file* exists and is a regular file.
- g *file* True if *file* exists and its set group ID flag is set.

-h file True if *file* exists and is a symbolic link.

-k file True if *file* exists and its sticky bit is set.

-n string True if the length of *string* is nonzero.

-p file True if *file* is a named pipe (FIFO).

-r file True if *file* exists and is readable.

-s file True if *file* exists and has a size greater than zero.

-t file_descriptor True if the file whose file descriptor number is *file_descriptor* is open and is associated with a terminal.

-u file True if *file* exists and its set user ID flag is set.

-w file True if *file* exists and is writable. True indicates only that the write flag is on. The file is not writable on a read-only file system even if this test indicates true.

-x file True if *file* exists and is executable. True indicates only that the execute flag is on. If *file* is a directory, true indicates that *file* can be searched.

-z string True if the length of *string* is zero.

-L file True if *file* exists and is a symbolic link. This operator is retained for compatibility with previous versions of this program. Do not rely on its existence; use **-h** instead.

-o file True if *file* exists and its owner matches the effective user id of this process.

-G file True if *file* exists and its group matches the effective group id of this process.

-s file True if *file* exists and is a socket.

file1 -nt file2 True if *file1* and *file2* exist and *file1* is newer than *file2*.

file1 -ot file2 True if *file1* and *file2* exist and *file1* is older than *file2*.

file1 -ef file2 True if *file1* and *file2* exist and refer to the same file.

string True if *string* is not the null string.

s1 = s2 True if the strings *s1* and *s2* are identical.

s1 != s2 True if the strings *s1* and *s2* are not identical.

s1 < s2 True if string *s1* comes before *s2* based on the ASCII value of their characters.

s1 > s2 True if string *s1* comes after *s2* based on the ASCII value of their characters.

n1 -eq n2 True if the integers *n1* and *n2* are algebraically equal.

n1 -ne n2 True if the integers *n1* and *n2* are not algebraically equal.

n1 -gt n2 True if the integer *n1* is algebraically greater than the integer *n2*.

n1 -ge n2 True if the integer *n1* is algebraically greater than or equal to the integer *n2*.

n1 -lt n2 True if the integer *n1* is algebraically less than the integer *n2*.

n1 -le n2 True if the integer *n1* is algebraically less than or equal to the integer *n2*.

These primaries can be combined with the following operators:

! *expression*
True if *expression* is false.

expression1 -a expression2
True if both *expression1* and *expression2* are true.

expression1 -o expression2
True if either *expression1* or *expression2* are true.

(*expression***)**
True if *expression* is true.

The **-a** operator has higher precedence than the **-o** operator.

times Print the accumulated user and system times for the shell and for processes run from the shell. The return status is 0.

trap [*action signal . . .*]

Cause the shell to parse and execute action when any of the specified signals are received. The signals are specified by signal number or as the name of the signal. If *signal* is 0 or EXIT, the action is executed when the shell exits. *action* may be empty (''), which causes the specified signals to be ignored. With *action* omitted or set to '-' the specified signals are set to their default action. When the shell forks off a subshell, it resets trapped (but not ignored) signals to the default action. The **trap** command has no effect on signals that were ignored on entry to the shell. **trap** without any arguments cause it to write a list of signals and their associated action to the standard output in a format that is suitable as an input to the shell that achieves the same trapping results.

Examples:

trap

List trapped signals and their corresponding action

trap '' INT QUIT tstop 30

Ignore signals INT QUIT TSTOP USR1

trap date INT

Print date upon receiving signal INT

type [*name . . .*]

Interpret each name as a command and print the resolution of the command search. Possible resolutions are: shell keyword, alias, shell builtin, command, tracked alias and not found. For aliases the alias expansion is printed; for commands and tracked aliases the complete pathname of the command is printed.

ulimit [**-H** | **-S**] [**-a** | **-tfdscmlpnv** [*value*]]

Inquire about or set the hard or soft limits on processes or set new limits. The choice between hard limit (which no process is allowed to violate, and which may not be raised once it has been lowered) and soft limit (which causes processes to be signaled but not necessarily killed, and which may be raised) is made with these flags:

- H** set or inquire about hard limits
- S** set or inquire about soft limits. If neither **-H** nor **-S** is specified, the soft limit is displayed or both limits are set. If both are specified, the last one wins.

The limit to be interrogated or set, then, is chosen by specifying any one of these flags:

- a** show all the current limits
- t** show or set the limit on CPU time (in seconds)
- f** show or set the limit on the largest file that can be created (in 512-byte blocks)
- d** show or set the limit on the data segment size of a process (in kilobytes)
- s** show or set the limit on the stack size of a process (in kilobytes)
- c** show or set the limit on the largest core dump size that can be produced (in 512-byte blocks)
- m** show or set the limit on the total physical memory that can be in use by a process (in kilobytes)
- l** show or set the limit on how much memory a process can lock with `mlock(2)` (in kilobytes)
- p** show or set the limit on the number of processes this user can have at one time
- n** show or set the limit on the number files a process can have open at once
- v** show or set the limit on the total virtual memory that can be in use by a process (in kilobytes)
- r** show or set the limit on the real-time scheduling priority of a process

If none of these is specified, it is the limit on file size that is shown or set. If value is specified, the limit is set to that number; otherwise the current limit is displayed.

Limits of an arbitrary process can be displayed or set using the `sysctl(8)` utility.

umask [*mask*]

Set the value of umask (see `umask(2)`) to the specified octal value. If the argument is omitted, the umask value is printed.

unalias [-a] [*name*]

If *name* is specified, the shell removes that alias. If **-a** is specified, all aliases are removed.

unset [-fv] [*name* . . .]

The specified variables and functions are unset and unexported. If **-f** or **-v** is specified, the corresponding function or variable is unset, respectively. If a given name corresponds to both a variable and a function, and no options are given, only the variable is unset.

wait [*job*]

Wait for the specified job to complete and return the exit status of the last process in the job. If the argument is omitted, wait for all jobs to complete and return an exit status of zero.

Command Line Editing

When `dash` is being used interactively from a terminal, the current command and the command history (see `fc` in **Builtins**) can be edited using vi-mode command-line editing. This mode uses commands, described below, similar to a subset of those described in the vi man page. The command `set -o vi` enables vi-mode editing and places sh into vi insert mode. With vi-mode enabled, sh can be switched between insert mode and command mode. It is similar to vi: typing `<ESC>` enters vi command mode. Hitting `<return>` while

in command mode will pass the line to the shell.

EXIT STATUS

Errors that are detected by the shell, such as a syntax error, will cause the shell to exit with a non-zero exit status. If the shell is not an interactive shell, the execution of the shell file will be aborted. Otherwise the shell will return the exit status of the last command executed, or if the exit builtin is used with a numeric argument, it will return the argument.

ENVIRONMENT

HOME	Set automatically by <code>login(1)</code> from the user's login directory in the password file (<code>passwd(4)</code>). This environment variable also functions as the default argument for the <code>cd</code> builtin.
PATH	The default search path for executables. See the above section Path Search .
CDPATH	The search path used with the <code>cd</code> builtin.
MAIL	The name of a mail file, that will be checked for the arrival of new mail. Overridden by <code>MAILPATH</code> .
MAILCHECK	The frequency in seconds that the shell checks for the arrival of mail in the files specified by the <code>MAILPATH</code> or the <code>MAIL</code> file. If set to 0, the check will occur at each prompt.
MAILPATH	A colon ":" separated list of file names, for the shell to check for incoming mail. This environment setting overrides the <code>MAIL</code> setting. There is a maximum of 10 mailboxes that can be monitored at once.
PS1	The primary prompt string, which defaults to "\$ ", unless you are the superuser, in which case it defaults to "# ".
PS2	The secondary prompt string, which defaults to ">".
PS4	Output before each line when execution trace (set -x) is enabled, defaults to "+".
IFS	Input Field Separators. This is normally set to <space>, <tab>, and <newline>. See the White Space Splitting section for more details.
TERM	The default terminal setting for the shell. This is inherited by children of the shell, and is used in the history editing modes.
HISTSIZE	The number of lines in the history buffer for the shell.
PWD	The logical value of the current working directory. This is set by the <code>cd</code> command.
OLDPWD	The previous logical value of the current working directory. This is set by the <code>cd</code> command.
PPID	The process ID of the parent process of the shell.

FILES

`$HOME/.profile`
`/etc/profile`

SEE ALSO

`csh(1)`, `echo(1)`, `getopt(1)`, `ksh(1)`, `login(1)`, `printf(1)`, `test(1)`, `getopt(3)`, `passwd(5)`, `environ(7)`, `sysctl(8)`

HISTORY

dash is a POSIX-compliant implementation of /bin/sh that aims to be as small as possible. **dash** is a direct descendant of the NetBSD version of ash (the Almquist SHell), ported to Linux in early 1997. It was renamed to **dash** in 2002.

BUGS

Setuid shell scripts should be avoided at all costs, as they are a significant security risk.

PS1, PS2, and PS4 should be subject to parameter expansion before being displayed.

NAME

sha1sum – compute and check SHA1 message digest

SYNOPSIS

sha1sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check SHA1 (160-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read SHA1 sums from the FILEs and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in FIPS-180-1. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

BUGS

Do not use the SHA-1 algorithm for security related purposes. Instead, use an SHA-2 algorithm, implemented in the programs sha224sum(1), sha256sum(1), sha384sum(1), sha512sum(1), or the BLAKE2 algorithm, implemented in b2sum(1)

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/sha1sum>>
or available locally via: info '(coreutils) sha1sum invocation'

NAME

sha224sum – compute and check SHA224 message digest

SYNOPSIS

sha224sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check SHA224 (224-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read SHA224 sums from the FILES and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in RFC 3874. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/sha224sum>>
or available locally via: info '(coreutils) sha2 utilities'

NAME

sha256sum – compute and check SHA256 message digest

SYNOPSIS

sha256sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check SHA256 (256-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read SHA256 sums from the FILES and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in FIPS-180-2. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/sha256sum>>
or available locally via: info '(coreutils) sha2 utilities'

NAME

sha384sum – compute and check SHA384 message digest

SYNOPSIS

sha384sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check SHA384 (384-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read SHA384 sums from the FILES and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in FIPS-180-2. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/sha384sum>>
or available locally via: info '(coreutils) sha2 utilities'

NAME

sha512sum – compute and check SHA512 message digest

SYNOPSIS

sha512sum [*OPTION*]... [*FILE*]...

DESCRIPTION

Print or check SHA512 (512-bit) checksums.

With no FILE, or when FILE is `-`, read standard input.

-b, --binary

read in binary mode

-c, --check

read SHA512 sums from the FILES and check them

--tag create a BSD-style checksum

-t, --text

read in text mode (default)

-z, --zero

end each output line with NUL, not newline, and disable file name escaping

The following five options are useful only when verifying checksums:

--ignore-missing

don't fail or report status for missing files

--quiet

don't print OK for each successfully verified file

--status

don't output anything, status code shows success

--strict

exit non-zero for improperly formatted checksum lines

-w, --warn

warn about improperly formatted checksum lines

--help display this help and exit

--version

output version information and exit

The sums are computed as described in FIPS-180-2. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a space, a character indicating input mode ('*' for binary, ' ' for text or where binary is insignificant), and name for each FILE.

Note: There is no difference between binary mode and text mode on GNU systems.

AUTHOR

Written by Ulrich Drepper, Scott Miller, and David Madore.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/sha512sum>>
or available locally via: info '(coreutils) sha2 utilities'

NAME

shadow – shadowed password file

DESCRIPTION

shadow is a file which contains the password information for the system's accounts and optional aging information.

This file must not be readable by regular users if password security is to be maintained.

Each line of this file contains 9 fields, separated by colons (“：“), in the following order:

login name

It must be a valid account name, which exist on the system.

encrypted password

This field may be empty, in which case no passwords are required to authenticate as the specified login name. However, some applications which read the /etc/shadow file may decide not to permit any access at all if the password field is empty.

A password field which starts with an exclamation mark means that the password is locked. The remaining characters on the line represent the password field before the password was locked.

Refer to **crypt(3)** for details on how this string is interpreted.

If the password field contains some string that is not a valid result of **crypt(3)**, for instance ! or *, the user will not be able to use a unix password to log in (but the user may log in the system by other means).

date of last password change

The date of the last password change, expressed as the number of days since Jan 1, 1970.

The value 0 has a special meaning, which is that the user should change her password the next time she will log in the system.

An empty field means that password aging features are disabled.

minimum password age

The minimum password age is the number of days the user will have to wait before she will be allowed to change her password again.

An empty field and value 0 mean that there are no minimum password age.

maximum password age

The maximum password age is the number of days after which the user will have to change her password.

After this number of days is elapsed, the password may still be valid. The user should be asked to change her password the next time she will log in.

An empty field means that there are no maximum password age, no password warning period, and no password inactivity period (see below).

If the maximum password age is lower than the minimum password age, the user cannot change her password.

password warning period

The number of days before a password is going to expire (see the maximum password age above) during which the user should be warned.

An empty field and value 0 mean that there are no password warning period.

password inactivity period

The number of days after a password has expired (see the maximum password age above) during which the password should still be accepted (and the user should update her password during the next login).

After expiration of the password and this expiration period is elapsed, no login is possible using the current user's password. The user should contact her administrator.

An empty field means that there are no enforcement of an inactivity period.

account expiration date

The date of expiration of the account, expressed as the number of days since Jan 1, 1970.

Note that an account expiration differs from a password expiration. In case of an account expiration, the user shall not be allowed to login. In case of a password expiration, the user is not allowed to login using her password.

An empty field means that the account will never expire.

The value 0 should not be used as it is interpreted as either an account with no expiration, or as an expiration on Jan 1, 1970.

reserved field

This field is reserved for future use.

FILES

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

/etc/shadow-

Backup file for /etc/shadow.

Note that this file is used by the tools of the shadow toolsuite, but not by all user and password management tools.

SEE ALSO

[chage\(1\)](#), [login\(1\)](#), [passwd\(1\)](#), [passwd\(5\)](#), [pwck\(8\)](#), [pwconv\(8\)](#), [pwunconv\(8\)](#), [su\(1\)](#), [sulogin\(8\)](#).

NAME

`shasum` – Print or Check SHA Checksums

SYNOPSIS

```
Usage: shasum [OPTION]... [FILE]...
Print or check SHA checksums.
With no FILE, or when FILE is -, read standard input.

-a, --algorithm    1 (default), 224, 256, 384, 512, 512224, 512256
-b, --binary       read in binary mode
-c, --check        read SHA sums from the FILES and check them
--tag              create a BSD-style checksum
-t, --text         read in text mode (default)
-U, --UNIVERSAL   read in Universal Newlines mode
                  produces same digest on Windows/Unix/Mac
-O, --01           read in BITS mode
                  ASCII '0' interpreted as 0-bit,
                  ASCII '1' interpreted as 1-bit,
                  all other characters ignored

The following five options are useful only when verifying checksums:
--ignore-missing  don't fail or report status for missing files
-q, --quiet        don't print OK for each successfully verified file
-s, --status       don't output anything, status code shows success
--strict          exit non-zero for improperly formatted checksum lines
-w, --warn         warn about improperly formatted checksum lines

-h, --help          display this help and exit
-v, --version      output version information and exit
```

When verifying SHA-512/224 or SHA-512/256 checksums, indicate the algorithm explicitly using the `-a` option, e.g.

```
shasum -a 512224 -c checksumfile
```

The sums are computed as described in FIPS PUB 180-4. When checking, the input should be a former output of this program. The default mode is to print a line with checksum, a character indicating type (`*` for binary, ` ` for text, `U` for UNIVERSAL, `^` for BITS), and name for each FILE. The line starts with a `\' character if the FILE name contains either newlines or backslashes, which are then replaced by the two-character sequences `\'n` and `\\` respectively.

Report `shasum` bugs to mshelor@cpan.org

DESCRIPTION

Running `shasum` is often the quickest way to compute SHA message digests. The user simply feeds data to the script through files or standard input, and then collects the results from standard output.

The following command shows how to compute digests for typical inputs such as the NIST test vector “abc”:

```
perl -e "print qq(abc)" | shasum
```

Or, if you want to use SHA-256 instead of the default SHA-1, simply say:

```
perl -e "print qq(abc)" | shasum -a 256
```

Since `shasum` mimics the behavior of the combined GNU `sha1sum`, `sha224sum`, `sha256sum`, `sha384sum`,

and *sha512sum* programs, you can install this script as a convenient drop-in replacement.

Unlike the GNU programs, *shasum* encompasses the full SHA standard by allowing partial-byte inputs. This is accomplished through the BITS option (*-0*). The following example computes the SHA-224 digest of the 7-bit message *0001100*:

```
perl -e "print qq(0001100)" | shasum -0 -a 224
```

AUTHOR

Copyright (C) 2003–2018 Mark Shelor <mshelor@cpan.org>.

SEE ALSO

shasum is implemented using the Perl module Digest::SHA.

NAME

shell-quote – quote arguments for safe use, unmodified in a shell command

SYNOPSIS

shell-quote [*switch*]... *arg*...

DESCRIPTION

shell-quote lets you pass arbitrary strings through the shell so that they won't be changed by the shell. This lets you process commands or files with embedded white space or shell globbing characters safely. Here are a few examples.

EXAMPLES**ssh preserving args**

When running a remote command with ssh, ssh doesn't preserve the separate arguments it receives. It just joins them with spaces and passes them to \$SHELL -c. This doesn't work as intended:

```
ssh host touch 'hi there'           # fails
```

It creates 2 files, *hi* and *there*. Instead, do this:

```
cmd='shell-quote touch \'hi there\'`  
ssh host "$cmd"
```

This gives you just 1 file, *hi there*.

process find output

It's not ordinarily possible to process an arbitrary list of files output by **find** with a shell script. Anything you put in \$IFS to split up the output could legitimately be in a file's name. Here's how you can do it using **shell-quote**:

```
eval set -- `find -type f -print0 | xargs -0 shell-quote --`
```

debug shell scripts

shell-quote is better than **echo** for debugging shell scripts.

```
debug() {  
    [ -z "$debug" ] || shell-quote "debug: \"\$@\""  
}
```

With **echo** you can't tell the difference between debug 'foo bar' and debug foo bar, but with **shell-quote** you can.

save a command for later

shell-quote can be used to build up a shell command to run later. Say you want the user to be able to give you switches for a command you're going to run. If you don't want the switches to be re-evaluated by the shell (which is usually a good idea, else there are things the user can't pass through), you can do something like this:

```
user_switches=  
while [ $# != 0 ]  
do  
    case x$1 in  
        x--pass-through)  
            [ $# -gt 1 ] || die "need an argument for $1"  
            user_switches="$user_switches `shell-quote -- \"$2\"`  
            shift;;  
        # process other switches  
    esac  
    shift  
done  
# later  
eval "shell-quote some-command $user_switches my args"
```

OPTIONS**--debug**

Turn debugging on.

--help

Show the usage message and die.

--version

Show the version number and exit.

AVAILABILITY

The code is licensed under the GNU GPL. Check <http://www.argon.org/~roderick/> or CPAN for updated versions.

AUTHOR

Roderick Schertler <roderick@argon.org>

NAME

shells – pathnames of valid login shells

DESCRIPTION

/etc/shells is a text file which contains the full pathnames of valid login shells. This file is consulted by **chsh(1)** and available to be queried by other programs.

Be aware that there are programs which consult this file to find out if a user is a normal user; for example, FTP daemons traditionally disallow access to users with shells not included in this file.

FILES

/etc/shells

EXAMPLES

/etc/shells may contain the following paths:

/bin/sh
/bin/bash
/bin/csh

SEE ALSO

chsh(1), **getusershell(3)**, **pam_shells(8)**

showmount(8) - Linux man page

Name

showmount - show mount information for an NFS server

Synopsis

```
showmount [ -adehv ] [ --all ] [ --directories ] [ --exports ] [ --help ] [ --version ]  
[ host ]
```

Description

showmount queries the mount daemon on a remote host for information about the state of the NFS server on that machine. With no options **showmount** lists the set of clients who are mounting from that host. The output from **showmount** is designed to appear as though it were processed through "sort -u".

Options

-a or **--all**

List both the client hostname or IP address and mounted directory in host:dir format. This info should not be considered reliable. See the notes on rmtab in [**rpc.mountd\(8\)**](#).

-d or **--directories**

List only the directories mounted by some client.

-e or **--exports**

Show the NFS server's export list.

-h or **--help**

Provide a short help summary.

-v or **--version**

Report the current version number of the program.

--no-headers

Suppress the descriptive headings from the output.

See Also

[**rpc.mountd\(8\)**](#), [**rpc.nfsd\(8\)**](#)

Bugs

The completeness and accuracy of the information that **showmount** displays varies according to the NFS server's implementation.

Because **showmount** sorts and unqs the output, it is impossible to determine from the output whether a client is mounting the same directory more than once.

Author

Rick Sladkey <jrs@world.std.com>

Referenced By

[**exports**\(5\)](#), [**fixmount**\(8\)](#), [**mountd**\(8\)](#), [**showmount_selinux**\(8\)](#)

NAME

skill, **snice** – send a signal or report process status

SYNOPSIS

skill [*signal*] [*options*] *expression*
snice [*new priority*] [*options*] *expression*

DESCRIPTION

These tools are obsolete and unportable. The command syntax is poorly defined. Consider using the killall, pkill, and pgrep commands instead.

The default signal for skill is TERM. Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: -9 –SIGKILL –KILL.

The default priority for snice is +4. Priority numbers range from +20 (slowest) to -20 (fastest). Negative priority numbers are restricted to administrative users.

OPTIONS**-f, --fast**

Fast mode. This option has not been implemented.

-i, --interactive

Interactive use. You will be asked to approve each action.

-l, --list

List all signal names.

-L, --table

List all signal names in a nice table.

-n, --no-action

No action; perform a simulation of events that would occur but do not actually change the system.

-v, --verbose

Verbose; explain what is being done.

-w, --warnings

Enable warnings. This option has not been implemented.

-h, --help

Display help text and exit.

-V, --version

Display version information.

PROCESS SELECTION OPTIONS

Selection criteria can be: terminal, user, pid, command. The options below may be used to ensure correct interpretation.

-t, --tty *tty*

The next expression is a terminal (tty or pty).

-u, --user *user*

The next expression is a username.

-p, --pid *pid*

The next expression is a process ID number.

-c, --command *command*

The next expression is a command name.

--ns *pid*

Match the processes that belong to the same namespace as pid.

--nslist ns,...

list which namespaces will be considered for the --ns option. Available namespaces: ipc, mnt, net, pid, user, uts.

SIGNALS

The behavior of signals is explained in **signal(7)** manual page.

EXAMPLES

snice -c seti -c crack +7

Slow down seti and crack commands.

skill -KILL -t /dev/pts/*

Kill users on PTY devices.

skill -STOP -u viro -u lm -u davem

Stop three users.

SEE ALSO

kill(1), kill(2), killall(1), nice(1), pkill(1), renice(1), signal(7)

STANDARDS

No standards apply.

AUTHOR

Albert Cahalan <albert@users.sf.net> wrote skill and snice in 1999 as a replacement for a non-free version.

REPORTING BUGS

Please send bug reports to <procps@freelists.org>

NAME

skill, **snice** – send a signal or report process status

SYNOPSIS

skill [*signal*] [*options*] *expression*
snice [*new priority*] [*options*] *expression*

DESCRIPTION

These tools are obsolete and unportable. The command syntax is poorly defined. Consider using the killall, pkill, and pgrep commands instead.

The default signal for skill is TERM. Use -l or -L to list available signals. Particularly useful signals include HUP, INT, KILL, STOP, CONT, and 0. Alternate signals may be specified in three ways: -9 –SIGKILL –KILL.

The default priority for snice is +4. Priority numbers range from +20 (slowest) to -20 (fastest). Negative priority numbers are restricted to administrative users.

OPTIONS**-f, --fast**

Fast mode. This option has not been implemented.

-i, --interactive

Interactive use. You will be asked to approve each action.

-l, --list

List all signal names.

-L, --table

List all signal names in a nice table.

-n, --no-action

No action; perform a simulation of events that would occur but do not actually change the system.

-v, --verbose

Verbose; explain what is being done.

-w, --warnings

Enable warnings. This option has not been implemented.

-h, --help

Display help text and exit.

-V, --version

Display version information.

PROCESS SELECTION OPTIONS

Selection criteria can be: terminal, user, pid, command. The options below may be used to ensure correct interpretation.

-t, --tty *tty*

The next expression is a terminal (tty or pty).

-u, --user *user*

The next expression is a username.

-p, --pid *pid*

The next expression is a process ID number.

-c, --command *command*

The next expression is a command name.

--ns *pid*

Match the processes that belong to the same namespace as pid.

--nslist ns,...

list which namespaces will be considered for the --ns option. Available namespaces: ipc, mnt, net, pid, user, uts.

SIGNALS

The behavior of signals is explained in **signal(7)** manual page.

EXAMPLES

snice -c seti -c crack +7

Slow down seti and crack commands.

skill -KILL -t /dev/pts/*

Kill users on PTY devices.

skill -STOP -u viro -u lm -u davem

Stop three users.

SEE ALSO

kill(1), kill(2), killall(1), nice(1), pkill(1), renice(1), signal(7)

STANDARDS

No standards apply.

AUTHOR

Albert Cahalan <albert@users.sf.net> wrote skill and snice in 1999 as a replacement for a non-free version.

REPORTING BUGS

Please send bug reports to <procps@freelists.org>

NAME

sort – sort lines of text files

SYNOPSIS

```
sort [OPTION]... [FILE]...
sort [OPTION]... --files0-from=F
```

DESCRIPTION

Write sorted concatenation of all FILE(s) to standard output.

With no FILE, or when FILE is **-**, read standard input.

Mandatory arguments to long options are mandatory for short options too. Ordering options:

-b, --ignore-leading-blanks

ignore leading blanks

-d, --dictionary-order

consider only blanks and alphanumeric characters

-f, --ignore-case

fold lower case to upper case characters

-g, --general-numeric-sort

compare according to general numerical value

-i, --ignore-nonprinting

consider only printable characters

-M, --month-sort

compare (unknown) < 'JAN' < ... < 'DEC'

-h, --human-numeric-sort

compare human readable numbers (e.g., 2K 1G)

-n, --numeric-sort

compare according to string numerical value

-R, --random-sort

shuffle, but group identical keys. See shuf(1)

--random-source=FILE

get random bytes from FILE

-r, --reverse

reverse the result of comparisons

--sort=WORD

sort according to WORD: general-numeric **-g**, human-numeric **-h**, month **-M**, numeric **-n**, random **-R**, version **-V**

-V, --version-sort

natural sort of (version) numbers within text

Other options:

--batch-size=NERGE

merge at most NERGE inputs at once; for more use temp files

-c, --check, --check=diagnose-first

check for sorted input; do not sort

-C, --check=quiet, --check=silent

like **-c**, but do not report first bad line

--compress-program=PROG

compress temporaries with PROG; decompress them with PROG **-d**

--debug
 annotate the part of the line used to sort, and warn about questionable usage to stderr

--files0-from=F
 read input from the files specified by NUL-terminated names in file F; If F is – then read names from standard input

-k, --key=KEYDEF
 sort via a key; KEYDEF gives location and type

-m, --merge
 merge already sorted files; do not sort

-o, --output=FILE
 write result to FILE instead of standard output

-s, --stable
 stabilize sort by disabling last-resort comparison

-S, --buffer-size=SIZE
 use SIZE for main memory buffer

-t, --field-separator=SEP
 use SEP instead of non-blank to blank transition

-T, --temporary-directory=DIR
 use DIR for temporaries, not \$TMPDIR or /tmp; multiple options specify multiple directories

--parallel=N
 change the number of sorts run concurrently to N

-u, --unique
 with -c, check for strict ordering; without -c, output only the first of an equal run

-z, --zero-terminated
 line delimiter is NUL, not newline

--help display this help and exit

--version
 output version information and exit

KEYDEF is F[.C][OPTS][,F[.C][OPTS]] for start and stop position, where F is a field number and C a character position in the field; both are origin 1, and the stop position defaults to the line's end. If neither -t nor -b is in effect, characters in a field are counted from the beginning of the preceding whitespace. OPTS is one or more single-letter ordering options [bdfgiMhnRrV], which override global ordering options for that key. If no key is given, use the entire line as the key. Use--deb ug to diagnose incorrect key usage.

SIZE may be followed by the following multiplicative suffixes: % 1% of memory, b 1, K 1024 (default), and so on for M, G, T, P, E, Z, Y.

*** WARNING *** The locale specified by the environment affects sort order. Set LC_ALL=C to get the traditional sort order that uses native byte values.

AUTHOR

Written by Mike Haertel and Paul Eggert.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
 Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent

permitted by law.

SEE ALSO

shuf(1), uniq(1)

Full documentation <<https://www.gnu.org/software/coreutils/sort>>
or available locally via: info '(coreutils) sort invocation'

NAME

ss – another utility to investigate sockets

SYNOPSIS

ss [options] [FILTER]

DESCRIPTION

ss is used to dump socket statistics. It allows showing information similar to *netstat*. It can display more TCP and state information than other tools.

OPTIONS

When no option is used **ss** displays a list of open non-listening sockets (e.g. TCP/UNIX/UDP) that have established connection.

-h, --help

Show summary of options.

-V, --version

Output version information.

-H, --no-header

Suppress header line.

-O, --oneline

Print each socket's data on a single line.

-n, --numeric

Do not try to resolve service names. Show exact bandwidth values, instead of human-readable.

-r, --resolve

Try to resolve numeric address/ports.

-a, --all

Display both listening and non-listening (for TCP this means established connections) sockets.

-l, --listening

Display only listening sockets (these are omitted by default).

-o, --options

Show timer information. For TCP protocol, the output format is:

timer:(<timer_name>,<expire_time>,<retrans>)

<timer_name>

the name of the timer, there are five kind of timer names:

on : means one of these timers: TCP retrans timer, TCP early retrans timer and tail loss probe timer

keepalive: tcp keep alive timer

timewait: timewait stage timer

persist: zero window probe timer

unknown: none of the above timers

<expire_time>

how long time the timer will expire

<retrans>

how many times the retransmission occurred

-e, --extended

Show detailed socket information. The output format is:

uid:<uid_number> ino:<inode_number> sk:<cookie>

<uid_number>
 the user id the socket belongs to

<inode_number>
 the socket's inode number in VFS

<cookie>
 an uuid of the socket

-m, --memory
 Show socket memory usage. The output format is:
 skmem:(r<rmem_alloc>,rb<recv_buf>,t<wmem_alloc>,tb<snd_buf>,
 f<fwd_alloc>,w<wmem_queued>,o<opt_mem>,
 bl<back_log>,d<sock_drop>)

<rmem_alloc>
 the memory allocated for receiving packet

<recv_buf>
 the total memory can be allocated for receiving packet

<wmem_alloc>
 the memory used for sending packet (which has been sent to layer 3)

<snd_buf>
 the total memory can be allocated for sending packet

<fwd_alloc>
 the memory allocated by the socket as cache, but not used for receiving/sending packet yet. If need memory to send/receive packet, the memory in this cache will be used before allocate additional memory.

<wmem_queued>
 The memory allocated for sending packet (which has not been sent to layer 3)

<ropt_mem>
 The memory used for storing socket option, e.g., the key for TCP MD5 signature

<back_log>
 The memory used for the sk backlog queue. On a process context, if the process is receiving packet, and a new packet is received, it will be put into the sk backlog queue, so it can be received by the process immediately

<sock_drop>
 the number of packets dropped before they are de-multiplexed into the socket

-p, --processes
 Show process using socket.

-i, --info
 Show internal TCP information. Below fields may appear:
 ts show string "ts" if the timestamp option is set
 sack show string "sack" if the sack option is set
 ecn show string "ecn" if the explicit congestion notification option is set
 ecnseen show string "ecnseen" if the saw ecn flag is found in received packets
 fastopen show string "fastopen" if the fastopen option is set

cong_alg
the congestion algorithm name, the default congestion algorithm is "cubic"

wscale:<snd_wscale>:<rcv_wscale>
if window scale option is used, this field shows the send scale factor and receive scale factor

rto:<icsk_rto>
tcp re-transmission timeout value, the unit is millisecond

backoff:<icsk_backoff>
used for exponential backoff re-transmission, the actual re-transmission timeout value is iks_rto << iks_backoff

rtt:<rtt>/<rttvar>
rtt is the average round trip time, rttvar is the mean deviation of rtt, their units are millisecond

ato:<ato>
ack timeout, unit is millisecond, used for delay ack mode

mss:<mss>
max segment size

cwnd:<cwnd>
congestion window size

pmtu:<pmtu>
path MTU value

ssthresh:<ssthresh>
tcp congestion window slow start threshold

bytes_acked:<bytes_acked>
bytes acked

bytes_received:<bytes_received>
bytes received

segs_out:<segs_out>
segments sent out

segs_in:<segs_in>
segments received

send <send_bps>bps
egress bps

lastsnd:<lastsnd>
how long time since the last packet sent, the unit is millisecond

lastrecv:<lastrecv>
how long time since the last packet received, the unit is millisecond

lastack:<lastack>
how long time since the last ack received, the unit is millisecond

pacing_rate <pacing_rate>bps/<max_pacing_rate>bps
the pacing rate and max pacing rate

recv_space:<recv_space>
a helper variable for TCP internal auto tuning socket receive buffer

tcp-ulp-mptcp flags:[MmBbJjecv] token:<rem_token(rem_id)/loc_token(loc_id)> seq:<sn> sfseq:<ssn> ssnoff:<off> maplen:<maplen>
MPTCP subflow information

- tos** Show ToS and priority information. Below fields may appear:
- tos** IPv4 Type-of-Service byte
 - tclass** IPv6 Traffic Class byte
 - class_id**
Class id set by net_cls cgroup. If class is zero this shows priority set by SO_PRIORITY.
- cgroup**
Show cgroup information. Below fields may appear:
- cgroup** Cgroup v2 pathname. This pathname is relative to the mount point of the hierarchy.
- K, --kill**
Attempts to forcibly close sockets. This option displays sockets that are successfully closed and silently skips sockets that the kernel does not support closing. It supports IPv4 and IPv6 sockets only.
- s, --summary**
Print summary statistics. This option does not parse socket lists obtaining summary from various sources. It is useful when amount of sockets is so huge that parsing /proc/net/tcp is painful.
- E, --events**
Continually display sockets as they are destroyed
- Z, --context**
As the **-p** option but also shows process security context.
- For **netlink(7)** sockets the initiating process context is displayed as follows:
1. If valid pid show the process context.
 2. If destination is kernel (pid = 0) show kernel initial context.
 3. If a unique identifier has been allocated by the kernel or netlink user, show context as "unavailable". This will generally indicate that a process has more than one netlink socket active.
- z, --contexts**
As the **-Z** option but also shows the socket context. The socket context is taken from the associated inode and is not the actual socket context held by the kernel. Sockets are typically labeled with the context of the creating process, however the context shown will reflect any policy role, type and/or range transition rules applied, and is therefore a useful reference.
- N NSNAME, --net=NSNAME**
Switch to the specified network namespace name.
- b, --bpf**
Show socket classic BPF filters (only administrators are allowed to get these information).
- 4, --ipv4**
Display only IP version 4 sockets (alias for -f inet).
- 6, --ipv6**
Display only IP version 6 sockets (alias for -f inet6).
- 0, --packet**
Display PACKET sockets (alias for -f link).
- t, --tcp**
Display TCP sockets.
- u, --udp**
Display UDP sockets.

-d, --dccp
Display DCCP sockets.

-w, --raw
Display RAW sockets.

-x, --unix
Display Unix domain sockets (alias for -f unix).

-S, --sctp
Display SCTP sockets.

--vsock
Display vsock sockets (alias for -f vsock).

--xdp Display XDP sockets (alias for -f xdp).

--inet-sockopt
Display inet socket options.

-f FAMILY, --family=FAMILY
Display sockets of type FAMILY. Currently the following families are supported: unix, inet, inet6, link, netlink, vssock, xdp.

-A QUERY, --query=QUERY, --socket=QUERY
List of socket tables to dump, separated by commas. The following identifiers are understood: all, inet, tcp, udp, raw, unix, packet, netlink, unix_dgram, unix_stream, unix_seqpacket, packet_raw, packet_dgram, dccp, sctp, vsock_stream, vsock_dgram, xdp Any item in the list may optionally be prefixed by an exclamation mark (!) to exclude that socket table from being dumped.

-D FILE, --diag=FILE
Do not display anything, just dump raw information about TCP sockets to FILE after applying filters. If FILE is - stdout is used.

-F FILE, --filter=FILE
Read filter information from FILE. Each line of FILE is interpreted like single command line option. If FILE is - stdin is used.

FILTER := [state STATE-FILTER] [EXPRESSION]
Please take a look at the official documentation for details regarding filters.

STATE-FILTER

STATE-FILTER allows to construct arbitrary set of states to match. Its syntax is sequence of keywords state and exclude followed by identifier of state.

Available identifiers are:

All standard TCP states: **established**, **syn-sent**, **syn-recv**, **fin-wait-1**, **fin-wait-2**, **time-wait**, **closed**, **close-wait**, **last-ack**, **listening** and **closing**.

all - for all the states

connected - all the states except for **listening** and **closed**

synchronized - all the **connected** states except for **syn-sent**

bucket - states, which are maintained as minisockets, i.e. **time-wait** and **syn-recv**

big - opposite to **bucket**

EXPRESSION

EXPRESSION allows filtering based on specific criteria. **EXPRESSION** consists of a series of predicates combined by boolean operators. The possible operators in increasing order of precedence are **or** (or | or ||), **and** (or & or &&), and **not** (or !). If no operator is between consecutive predicates, an implicit **and** operator is assumed. Subexpressions can be grouped with "(" and ")".

The following predicates are supported:

{dst|src} [=] HOST

Test if the destination or source matches HOST. See HOST SYNTAX for details.

{dport|sport} [OP] [FAMILY:]PORT

Compare the destination or source port to PORT. OP can be any of "<", "<=", "=", "!=" or ">". Following normal arithmetic rules. FAMILY and PORT are as described in HOST SYNTAX below.

dev [=!=] DEVICE

Match based on the device the connection uses. DEVICE can either be a device name or the index of the interface.

fwmark [=!=] MASK

Matches based on the fwmark value for the connection. This can either be a specific mark value or a mark value followed by a "/" and a bitmask of which bits to use in the comparison. For example "fwmark = 0x01/0x03" would match if the two least significant bits of the fwmark were 0x01.

cgroup [=!=] PATH

Match if the connection is part of a cgroup at the given path.

autobound

Match if the port or path of the source address was automatically allocated (rather than explicitly specified).

Most operators have aliases. If no operator is supplied "=" is assumed. Each of the following groups of operators are all equivalent:

- == eq
- != ne neq
- > gt
- < lt
- >= ge geq
- <= le leq
- ! not
- | || or
- & && and

HOST SYNTAX

The general host syntax is [FAMILY:]ADDRESS[:PORT].

FAMILY must be one of the families supported by the -f option. If not given it defaults to the family given with the -f option, and if that is also missing, will assume either inet or inet6. Note that all host conditions in the expression should either all be the same family or be only inet and inet6. If there is some other mixture of families, the results will probably be unexpected.

The form of ADDRESS and PORT depends on the family used. "*" can be used as a wildcard for either the address or port. The details for each family are as follows:

unix ADDRESS is a glob pattern (see **fnmatch(3)**) that will be matched case-insensitively against the unix socket's address. Both path and abstract names are supported. Unix addresses do not support

a port, and "*" cannot be used as a wildcard.

link ADDRESS is the case-insensitive name of an Ethernet protocol to match. PORT is either a device name or a device index for the desired link device, as seen in the output of ip link.

netlink ADDRESS is a descriptor of the netlink family. Possible values come from /etc/iproute2/nl_protos. PORT is the port id of the socket, which is usually the same as the owning process id. The value "kernel" can be used to represent the kernel (port id of 0).

vsock ADDRESS is an integer representing the CID address, and PORT is the port.

inet and inet6

ADDRESS is an ip address (either v4 or v6 depending on the family) or a DNS hostname that resolves to an ip address of the required version. An ipv6 address must be enclosed in "[" and "]" to disambiguate the port separator. The address may additionally have a prefix length given in CIDR notation (a slash followed by the prefix length in bits). PORT is either the numerical socket port, or the service name for the port to match.

USAGE EXAMPLES

ss -t -a Display all TCP sockets.

ss -t -a -Z

Display all TCP sockets with process SELinux security contexts.

ss -u -a Display all UDP sockets.

ss -o state established '(dport = :ssh or sport = :ssh)'

Display all established ssh connections.

ss -x src /tmp/.X11-unix/*

Find all local processes connected to X server.

ss -o state fin-wait-1 '(sport = :http or sport = :https)' dst 193.233.7/24

List all the tcp sockets in state FIN-WAIT-1 for our apache to network 193.233.7/24 and look at their timers.

ss -a -A 'all,!tcp'

List sockets in all states from all socket tables but TCP.

SEE ALSO

[ip\(8\)](#),

RFC 793 - <https://tools.ietf.org/rfc/rfc793.txt> (TCP states)

AUTHOR

ss was written by Alexey Kuznetsov, <kuznet@ms2.inr.ac.ru>.

This manual page was written by Michael Prokop <mika@grml.org> for the Debian project (but may be used by others).

NAME

ssh — OpenSSH remote login client

SYNOPSIS

```
ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
     [-c cipher_spec] [-D [bind_address:]port] [-E log_file] [-e escape_char]
     [-F configfile] [-I pkcs11] [-i identity_file] [-J destination]
     [-L address] [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option]
     [-p port] [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
     [-w local_tun[:remote_tun]] destination [command [argument ...]]
```

DESCRIPTION

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections, arbitrary TCP ports and UNIX-domain sockets can also be forwarded over the secure channel.

ssh connects and logs into the specified *destination*, which may be specified as either [user@]hostname or a URI of the form ssh://[user@]hostname[:port]. The user must prove their identity to the remote machine using one of several methods (see below).

If a *command* is specified, it will be executed on the remote host instead of a login shell. A complete command line may be specified as *command*, or it may have additional arguments. If supplied, the arguments will be appended to the command, separated by spaces, before it is sent to the server to be executed.

The options are as follows:

- 4** Forces **ssh** to use IPv4 addresses only.
- 6** Forces **ssh** to use IPv6 addresses only.
- A** Enables forwarding of connections from an authentication agent such as **ssh-agent(1)**. This can also be specified on a per-host basis in a configuration file.

Agent forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the agent's UNIX-domain socket) can access the local agent through the forwarded connection. An attacker cannot obtain key material from the agent, however they can perform operations on the keys that enable them to authenticate using the identities loaded into the agent. A safer alternative may be to use a jump host (see **-J**).

- a** Disables forwarding of the authentication agent connection.

-B bind_interface

Bind to the address of *bind_interface* before attempting to connect to the destination host. This is only useful on systems with more than one address.

-b bind_address

Use *bind_address* on the local machine as the source address of the connection. Only useful on systems with more than one address.

- C** Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11, TCP and UNIX-domain connections). The compression algorithm is the same used by **gzip(1)**. Compression is desirable on modem lines and other slow connections, but will only slow down things on fast networks. The default value can be set on a host-by-host basis in the configuration files; see the **Compression** option.

-c *cipher_spec*

Selects the cipher specification for encrypting the session. *cipher_spec* is a comma-separated list of ciphers listed in order of preference. See the **Ciphers** keyword in *ssh_config(5)* for more information.

-D [*bind_address*:]*port*

Specifies a local “dynamic” application-level port forwarding. This works by allocating a socket to listen to *port* on the local side, optionally bound to the specified *bind_address*. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine. Currently the SOCKS4 and SOCKS5 protocols are supported, and **ssh** will act as a SOCKS server. Only root can forward privileged ports. Dynamic port forwardings can also be specified in the configuration file.

IPv6 addresses can be specified by enclosing the address in square brackets. Only the superuser can forward privileged ports. By default, the local port is bound in accordance with the **GatewayPorts** setting. However, an explicit *bind_address* may be used to bind the connection to a specific address. The *bind_address* of “localhost” indicates that the listening port be bound for local use only, while an empty address or ‘*’ indicates that the port should be available from all interfaces.

-E *log_file*

Append debug logs to *log_file* instead of standard error.

-e *escape_char*

Sets the escape character for sessions with a pty (default: ‘~’). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (‘.’) closes the connection; followed by control-Z suspends the connection; and followed by itself sends the escape character once. Setting the character to “none” disables any escapes and makes the session fully transparent.

-F *configfile*

Specifies an alternative per-user configuration file. If a configuration file is given on the command line, the system-wide configuration file (*/etc/ssh/ssh_config*) will be ignored. The default for the per-user configuration file is *~/.ssh/config*. If set to “none”, no configuration files will be read.

-f Requests **ssh** to go to background just before command execution. This is useful if **ssh** is going to ask for passwords or passphrases, but the user wants it in the background. This implies **-n**. The recommended way to start X11 programs at a remote site is with something like **ssh -f host xterm**.

If the **ExitOnForwardFailure** configuration option is set to “yes”, then a client started with **-f** will wait for all remote port forwards to be successfully established before placing itself in the background. Refer to the description of **ForkAfterAuthentication** in *ssh_config(5)* for details.

-G Causes **ssh** to print its configuration after evaluating **Host** and **Match** blocks and exit.

-g Allows remote hosts to connect to local forwarded ports. If used on a multiplexed connection, then this option must be specified on the master process.

-I *pkcs11*

Specify the PKCS#11 shared library **ssh** should use to communicate with a PKCS#11 token providing keys for user authentication.

-i *identity_file*

Selects a file from which the identity (private key) for public key authentication is read. You can also specify a public key file to use the corresponding private key that is loaded in `ssh-agent(1)` when the private key file is not present locally. The default is `~/.ssh/id_rsa`, `~/.ssh/id_ecdsa`, `~/.ssh/id_ecdsa_sk`, `~/.ssh/id_ed25519`, `~/.ssh/id_ed25519_sk` and `~/.ssh/id_dsa`. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple **-i** options (and multiple identities specified in configuration files). If no certificates have been explicitly specified by the **CertificateFile** directive, **ssh** will also try to load certificate information from the filename obtained by appending `-cert.pub` to identity filenames.

-J *destination*

Connect to the target host by first making a **ssh** connection to the jump host described by *destination* and then establishing a TCP forwarding to the ultimate destination from there. Multiple jump hops may be specified separated by comma characters. This is a shortcut to specify a **ProxyJump** configuration directive. Note that configuration directives supplied on the command-line generally apply to the destination host and not any specified jump hosts. Use `~/.ssh/config` to specify configuration for jump hosts.

-K Enables GSSAPI-based authentication and forwarding (delegation) of GSSAPI credentials to the server.

-k Disables forwarding (delegation) of GSSAPI credentials to the server.

-L `[bind_address:]port:host:hostport`

-L `[bind_address:]port:remote_socket`

-L `local_socket:host:hostport`

-L `local_socket:remote_socket`

Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be forwarded to the given host and port, or Unix socket, on the remote side. This works by allocating a socket to listen to either a TCP *port* on the local side, optionally bound to the specified *bind_address*, or to a Unix socket. Whenever a connection is made to the local port or socket, the connection is forwarded over the secure channel, and a connection is made to either *host* port *hostport*, or the Unix socket *remote_socket*, from the remote machine.

Port forwardings can also be specified in the configuration file. Only the superuser can forward privileged ports. IPv6 addresses can be specified by enclosing the address in square brackets.

By default, the local port is bound in accordance with the **GatewayPorts** setting. However, an explicit *bind_address* may be used to bind the connection to a specific address. The *bind_address* of “localhost” indicates that the listening port be bound for local use only, while an empty address or ‘*’ indicates that the port should be available from all interfaces.

-l *login_name*

Specifies the user to log in as on the remote machine. This also may be specified on a per-host basis in the configuration file.

-M Places the **ssh** client into “master” mode for connection sharing. Multiple **-M** options places **ssh** into “master” mode but with confirmation required using `ssh-askpass(1)` before each operation that changes the multiplexing state (e.g. opening a new session). Refer to the description of **ControlMaster** in `ssh_config(5)` for details.

-m *mac_spec*

A comma-separated list of MAC (message authentication code) algorithms, specified in order of preference. See the **MACs** keyword for more information.

- N Do not execute a remote command. This is useful for just forwarding ports. Refer to the description of **SessionType** in **ssh_config(5)** for details.
- n Redirects stdin from /dev/null (actually, prevents reading from stdin). This must be used when **ssh** is run in the background. A common trick is to use this to run X11 programs on a remote machine. For example, **ssh -n shadows.cs.hut.fi emacs &** will start an emacs on shadows.cs.hut.fi, and the X11 connection will be automatically forwarded over an encrypted channel. The **ssh** program will be put in the background. (This does not work if **ssh** needs to ask for a password or passphrase; see also the **-f** option.) Refer to the description of **StdinNull** in **ssh_config(5)** for details.
- o *ctl_cmd*
Control an active connection multiplexing master process. When the **-o** option is specified, the *ctl_cmd* argument is interpreted and passed to the master process. Valid commands are: “check” (check that the master process is running), “forward” (request forwardings without command execution), “cancel” (cancel forwardings), “exit” (request the master to exit), and “stop” (request the master to stop accepting further multiplexing requests).
- o *option*
Can be used to give options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag. For full details of the options listed below, and their possible values, see **ssh_config(5)**.
 - AddKeysToAgent
 - AddressFamily
 - BatchMode
 - BindAddress
 - CanonicalDomains
 - CanonicalizeFallbackLocal
 - CanonicalizeHostname
 - CanonicalizeMaxDots
 - CanonicalizePermittedCNAMEs
 - CASignatureAlgorithms
 - CertificateFile
 - CheckHostIP
 - Ciphers
 - ClearAllForwardings
 - Compression
 - ConnectionAttempts
 - ConnectTimeout
 - ControlMaster
 - ControlPath
 - ControlPersist
 - DynamicForward
 - EscapeChar
 - ExitOnForwardFailure
 - FingerprintHash
 - ForkAfterAuthentication
 - ForwardAgent
 - ForwardX11
 - ForwardX11Timeout

ForwardX11Trusted
GatewayPorts
GlobalKnownHostsFile
GSSAPIAuthentication
GSSAPIKeyExchange
GSSAPIClientIdentity
GSSAPIDelegateCredentials
GSSAPIKexAlgorithms
GSSAPIRenewalForcesRekey
GSSAPIServerIdentity
GSSAPITrustDns
HashKnownHosts
Host
HostbasedAcceptedAlgorithms
HostbasedAuthentication
HostKeyAlgorithms
HostKeyAlias
Hostname
IdentitiesOnly
IdentityAgent
IdentityFile
IPQoS
KbdInteractiveAuthentication
KbdInteractiveDevices
KexAlgorithms
KnownHostsCommand
LocalCommand
LocalForward
LogLevel
MACs
Match
NoHostAuthenticationForLocalhost
NumberOfPasswordPrompts
PasswordAuthentication
PermitLocalCommand
PermitRemoteOpen
PKCS11Provider
Port
PreferredAuthentications
ProxyCommand
ProxyJump
ProxyUseFdpass
PubkeyAcceptedAlgorithms
PubkeyAuthentication
RekeyLimit
RemoteCommand
RemoteForward
RequestTTY
SendEnv

```
ServerAliveInterval  
ServerAliveCountMax  
SessionType  
SetEnv  
StdinNull  
StreamLocalBindMask  
StreamLocalBindUnlink  
StrictHostKeyChecking  
TCPKeepAlive  
Tunnel  
TunnelDevice  
UpdateHostKeys  
User  
UserKnownHostsFile  
VerifyHostKeyDNS  
VisualHostKey  
XAuthLocation
```

-p port

Port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.

-Q query_option

Queries for the algorithms supported by one of the following features: *cipher* (supported symmetric ciphers), *cipher-auth* (supported symmetric ciphers that support authenticated encryption), *help* (supported query terms for use with the **-Q** flag), *mac* (supported message integrity codes), *kex* (key exchange algorithms), *kex-gss* (GSSAPI key exchange algorithms), *key* (key types), *key-cert* (certificate key types), *key-plain* (non-certificate key types), *key-sig* (all key types and signature algorithms), *protocol-version* (supported SSH protocol versions), and *sig* (supported signature algorithms). Alternatively, any keyword from *ssh_config(5)* or *sshd_config(5)* that takes an algorithm list may be used as an alias for the corresponding *query_option*.

-q Quiet mode. Causes most warning and diagnostic messages to be suppressed.

```
-R [bind_address:]port:host:hostport  
-R [bind_address:]port:local_socket  
-R remote_socket:host:hostport  
-R remote_socket:local_socket  
-R [bind_address:]port
```

Specifies that connections to the given TCP port or Unix socket on the remote (server) host are to be forwarded to the local side.

This works by allocating a socket to listen to either a TCP *port* or to a Unix socket on the remote side. Whenever a connection is made to this port or Unix socket, the connection is forwarded over the secure channel, and a connection is made from the local machine to either an explicit destination specified by *host* *port* *hostport*, or *local_socket*, or, if no explicit destination was specified, **ssh** will act as a SOCKS 4/5 proxy and forward connections to the destinations requested by the remote SOCKS client.

Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine. IPv6 addresses can be specified by enclosing the address in square brackets.

By default, TCP listening sockets on the server will be bound to the loopback interface only. This may be overridden by specifying a *bind_address*. An empty *bind_address*, or the address ‘*’, indicates that the remote socket should listen on all interfaces. Specifying a remote *bind_address* will only succeed if the server’s **GatewayPorts** option is enabled (see *sshd_config(5)*).

If the *port* argument is ‘0’, the listen port will be dynamically allocated on the server and reported to the client at run time. When used together with **-O forward** the allocated port will be printed to the standard output.

-s *ctl_path*

Specifies the location of a control socket for connection sharing, or the string “none” to disable connection sharing. Refer to the description of **ControlPath** and **ControlMaster** in *ssh_config(5)* for details.

-s May be used to request invocation of a subsystem on the remote system. Subsystems facilitate the use of SSH as a secure transport for other applications (e.g. *sftp(1)*). The subsystem is specified as the remote command. Refer to the description of **SessionType** in *ssh_config(5)* for details.

-T Disable pseudo-terminal allocation.

-t Force pseudo-terminal allocation. This can be used to execute arbitrary screen-based programs on a remote machine, which can be very useful, e.g. when implementing menu services. Multiple **-t** options force tty allocation, even if **ssh** has no local tty.

-v Display the version number and exit.

-v Verbose mode. Causes **ssh** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Multiple **-v** options increase the verbosity. The maximum is 3.

-W *host:port*

Requests that standard input and output on the client be forwarded to *host* on *port* over the secure channel. Implies **-N**, **-T**, **ExitOnForwardFailure** and **ClearAllForwardings**, though these can be overridden in the configuration file or using **-o** command line options.

-w *local_tun[:remote_tun]*

Requests tunnel device forwarding with the specified tun(4) devices between the client (*local_tun*) and the server (*remote_tun*).

The devices may be specified by numerical ID or the keyword “any”, which uses the next available tunnel device. If *remote_tun* is not specified, it defaults to “any”. See also the **Tunnel** and **TunnelDevice** directives in *ssh_config(5)*.

If the **Tunnel** directive is unset, it will be set to the default tunnel mode, which is “point-to-point”. If a different **Tunnel** forwarding mode is desired, then it should be specified before **-w**.

-x Enables X11 forwarding. This can also be specified on a per-host basis in a configuration file.

X11 forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the user’s X authorization database) can access the local X11 display through the forwarded connection. An attacker may then be able to perform activities such as keystroke monitoring.

For this reason, X11 forwarding is subjected to X11 SECURITY extension restrictions by default. Refer to the **ssh -Y** option and the **ForwardX11Trusted** directive in *ssh_config(5)* for more information.

(Debian-specific: X11 forwarding is not subjected to X11 SECURITY extension restrictions by default, because too many programs currently crash in this mode. Set the **ForwardX11Trusted** option to “no” to restore the upstream behaviour. This may change in future depending on client-side improvements.)

- x Disables X11 forwarding.
 - y Enables trusted X11 forwarding. Trusted X11 forwardings are not subjected to the X11 SECURITY extension controls.
- (Debian-specific: In the default configuration, this option is equivalent to -x, since **ForwardX11Trusted** defaults to “yes” as described above. Set the **ForwardX11Trusted** option to “no” to restore the upstream behaviour. This may change in future depending on client-side improvements.)
- y Send log information using the `syslog(3)` system module. By default this information is sent to stderr.

ssh may additionally obtain configuration data from a per-user configuration file and a system-wide configuration file. The file format and configuration options are described in `ssh_config(5)`.

AUTHENTICATION

The OpenSSH SSH client supports SSH protocol 2.

The methods available for authentication are: GSSAPI-based authentication, host-based authentication, public key authentication, keyboard-interactive authentication, and password authentication. Authentication methods are tried in the order specified above, though **PreferredAuthentications** can be used to change the default order.

Host-based authentication works as follows: If the machine the user logs in from is listed in `/etc/hosts.equiv` or `/etc/ssh/shosts.equiv` on the remote machine, the user is non-root and the user names are the same on both sides, or if the files `~/.rhosts` or `~/.shosts` exist in the user’s home directory on the remote machine and contain a line containing the name of the client machine and the name of the user on that machine, the user is considered for login. Additionally, the server *must* be able to verify the client’s host key (see the description of `/etc/ssh/ssh_known_hosts` and `~/.ssh/known_hosts`, below) for login to be permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing. [Note to the administrator: `/etc/hosts.equiv`, `~/.rhosts`, and the rlogin/rsh protocol in general, are inherently insecure and should be disabled if security is desired.]

Public key authentication works as follows: The scheme is based on public-key cryptography, using cryptosystems where encryption and decryption are done using separate keys, and it is unfeasible to derive the decryption key from the encryption key. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. **ssh** implements public key authentication protocol automatically, using one of the DSA, ECDSA, Ed25519 or RSA algorithms. The HISTORY section of `ssl(8)` (on non-OpenBSD systems, see <http://www.openbsd.org/cgi-bin/man.cgi?query=ssl&sektion=8#HISTORY>) contains a brief discussion of the DSA and RSA algorithms.

The file `~/.ssh/authorized_keys` lists the public keys that are permitted for logging in. When the user logs in, the **ssh** program tells the server which key pair it would like to use for authentication. The client proves that it has access to the private key and the server checks that the corresponding public key is authorized to accept the account.

The server may inform the client of errors that prevented public key authentication from succeeding after authentication completes using a different method. These may be viewed by increasing the **LogLevel** to **DEBUG** or higher (e.g. by using the **-v** flag).

The user creates their key pair by running `ssh-keygen(1)`. This stores the private key in `~/.ssh/id_dsa` (DSA), `~/.ssh/id_ecdsa` (ECDSA), `~/.ssh/id_ecdsa_sk` (authenticator-hosted ECDSA), `~/.ssh/id_ed25519` (Ed25519), `~/.ssh/id_ed25519_sk` (authenticator-hosted Ed25519), or `~/.ssh/id_rsa` (RSA) and stores the public key in `~/.ssh/id_dsa.pub` (DSA), `~/.ssh/id_ecdsa.pub` (ECDSA), `~/.ssh/id_ecdsa_sk.pub` (authenticator-hosted ECDSA), `~/.ssh/id_ed25519.pub` (Ed25519), `~/.ssh/id_ed25519_sk.pub` (authenticator-hosted Ed25519), or `~/.ssh/id_rsa.pub` (RSA) in the user's home directory. The user should then copy the public key to `~/.ssh/authorized_keys` in their home directory on the remote machine. The `authorized_keys` file corresponds to the conventional `~/.rhosts` file, and has one key per line, though the lines can be very long. After this, the user can log in without giving the password.

A variation on public key authentication is available in the form of certificate authentication: instead of a set of public/private keys, signed certificates are used. This has the advantage that a single trusted certification authority can be used in place of many public/private keys. See the CERTIFICATES section of `ssh-keygen(1)` for more information.

The most convenient way to use public key or certificate authentication may be with an authentication agent. See `ssh-agent(1)` and (optionally) the **AddKeysToAgent** directive in `ssh_config(5)` for more information.

Keyboard-interactive authentication works as follows: The server sends an arbitrary "challenge" text and prompts for a response, possibly multiple times. Examples of keyboard-interactive authentication include BSD Authentication (see `login.conf(5)`) and PAM (some non-OpenBSD systems).

Finally, if other authentication methods fail, `ssh` prompts the user for a password. The password is sent to the remote host for checking; however, since all communications are encrypted, the password cannot be seen by someone listening on the network.

`ssh` automatically maintains and checks a database containing identification for all hosts it has ever been used with. Host keys are stored in `~/.ssh/known_hosts` in the user's home directory. Additionally, the file `/etc/ssh/ssh_known_hosts` is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification ever changes, `ssh` warns about this and disables password authentication to prevent server spoofing or man-in-the-middle attacks, which could otherwise be used to circumvent the encryption. The **StrictHostKeyChecking** option can be used to control logins to machines whose host key is not known or has changed.

When the user's identity has been accepted by the server, the server either executes the given command in a non-interactive session or, if no command has been specified, logs into the machine and gives the user a normal shell as an interactive session. All communication with the remote command or shell will be automatically encrypted.

If an interactive session is requested `ssh` by default will only request a pseudo-terminal (pty) for interactive sessions when the client has one. The flags **-T** and **-t** can be used to override this behaviour.

If a pseudo-terminal has been allocated the user may use the escape characters noted below.

If no pseudo-terminal has been allocated, the session is transparent and can be used to reliably transfer binary data. On most systems, setting the escape character to "none" will also make the session transparent even if a tty is used.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP connections have been closed.

ESCAPE CHARACTERS

When a pseudo-terminal has been requested, `ssh` supports a number of functions through the use of an escape character.

A single tilde character can be sent as `~` or by following the tilde by a character other than those described below. The escape character must always follow a newline to be interpreted as special. The escape character can be changed in configuration files using the **EscapeChar** configuration directive or on the command line by the **-e** option.

The supported escapes (assuming the default `~`) are:

- ~.** Disconnect.
- ~~z** Background **ssh**.
- ~#** List forwarded connections.
- ~&** Background **ssh** at logout when waiting for forwarded connection / X11 sessions to terminate.
- ~?** Display a list of escape characters.
- ~B** Send a BREAK to the remote system (only useful if the peer supports it).
- ~C** Open command line. Currently this allows the addition of port forwardings using the **-L**, **-R** and **-D** options (see above). It also allows the cancellation of existing port-forwardings with **-KL[bind_address:]port** for local, **-KR[bind_address:]port** for remote and **-KD[bind_address:]port** for dynamic port-forwardings. **!command** allows the user to execute a local command if the **PermitLocalCommand** option is enabled in **ssh_config(5)**. Basic help is available, using the **-h** option.
- ~R** Request rekeying of the connection (only useful if the peer supports it).
- ~v** Decrease the verbosity (**LogLevel**) when errors are being written to stderr.
- ~v** Increase the verbosity (**LogLevel**) when errors are being written to stderr.

TCP FORWARDING

Forwarding of arbitrary TCP connections over a secure channel can be specified either on the command line or in a configuration file. One possible application of TCP forwarding is a secure connection to a mail server; another is going through firewalls.

In the example below, we look at encrypting communication for an IRC client, even though the IRC server it connects to does not directly support encrypted communication. This works as follows: the user connects to the remote host using **ssh**, specifying the ports to be used to forward the connection. After that it is possible to start the program locally, and **ssh** will encrypt and forward the connection to the remote server.

The following example tunnels an IRC session from the client to an IRC server at “server.example.com”, joining channel “#users”, nickname “pinky”, using the standard IRC port, 6667:

```
$ ssh -f -L 6667:localhost:6667 server.example.com sleep 10
$ irc -c '#users' pinky IRC/127.0.0.1
```

The **-f** option backgrounds **ssh** and the remote command “sleep 10” is specified to allow an amount of time (10 seconds, in the example) to start the program which is going to use the tunnel. If no connections are made within the time specified, **ssh** will exit.

X11 FORWARDING

If the **ForwardX11** variable is set to “yes” (or see the description of the **-X**, **-x**, and **-Y** options above) and the user is using X11 (the **DISPLAY** environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set **DISPLAY**. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DISPLAY value set by **ssh** will point to the server machine, but with a display number greater than zero. This is normal, and happens because **ssh** creates a “proxy” X server on the server machine for forwarding the connections over the encrypted channel.

ssh will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

If the **ForwardAgent** variable is set to “yes” (or see the description of the **-A** and **-a** options above) and the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side.

VERIFYING HOST KEYS

When connecting to a server for the first time, a fingerprint of the server’s public key is presented to the user (unless the option **StrictHostKeyChecking** has been disabled). Fingerprints can be determined using **ssh-keygen(1)**:

```
$ ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
```

If the fingerprint is already known, it can be matched and the key can be accepted or rejected. If only legacy (MD5) fingerprints for the server are available, the **ssh-keygen(1)** **-E** option may be used to downgrade the fingerprint algorithm to match.

Because of the difficulty of comparing host keys just by looking at fingerprint strings, there is also support to compare host keys visually, using *random art*. By setting the **VisualHostKey** option to “yes”, a small ASCII graphic gets displayed on every login to a server, no matter if the session itself is interactive or not. By learning the pattern a known server produces, a user can easily find out that the host key has changed when a completely different pattern is displayed. Because these patterns are not unambiguous however, a pattern that looks similar to the pattern remembered only gives a good probability that the host key is the same, not guaranteed proof.

To get a listing of the fingerprints along with their random art for all known hosts, the following command line can be used:

```
$ ssh-keygen -lv -f ~/.ssh/known_hosts
```

If the fingerprint is unknown, an alternative method of verification is available: SSH fingerprints verified by DNS. An additional resource record (RR), SSHFP, is added to a zonefile and the connecting client is able to match the fingerprint with that of the key presented.

In this example, we are connecting a client to a server, “host.example.com”. The SSHFP resource records should first be added to the zonefile for host.example.com:

```
$ ssh-keygen -r host.example.com.
```

The output lines will have to be added to the zonefile. To check that the zone is answering fingerprint queries:

```
$ dig -t SSHFP host.example.com
```

Finally the client connects:

```
$ ssh -o "VerifyHostKeyDNS ask" host.example.com
[...]
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

See the **VerifyHostKeyDNS** option in **ssh_config(5)** for more information.

SSH-BASED VIRTUAL PRIVATE NETWORKS

ssh contains support for Virtual Private Network (VPN) tunnelling using the **tun(4)** network pseudo-device, allowing two networks to be joined securely. The **sshd_config(5)** configuration option **PermitTunnel** controls whether the server supports this, and at what level (layer 2 or 3 traffic).

The following example would connect client network 10.0.50.0/24 with remote network 10.0.99.0/24 using a point-to-point connection from 10.1.1.1 to 10.1.1.2, provided that the SSH server running on the gateway to the remote network, at 192.168.1.15, allows it.

On the client:

```
# ssh -f -w 0:1 192.168.1.15 true
# ifconfig tun0 10.1.1.1 10.1.1.2 netmask 255.255.255.252
# route add 10.0.99.0/24 10.1.1.2
```

On the server:

```
# ifconfig tun1 10.1.1.2 10.1.1.1 netmask 255.255.255.252
# route add 10.0.50.0/24 10.1.1.1
```

Client access may be more finely tuned via the **/root/.ssh/authorized_keys** file (see below) and the **PermitRootLogin** server option. The following entry would permit connections on **tun(4)** device 1 from user “jane” and on tun device 2 from user “john”, if **PermitRootLogin** is set to “forced-commands-only”:

```
tunnel="1",command="sh /etc/netstart tun1" ssh-rsa ... jane
tunnel="2",command="sh /etc/netstart tun2" ssh-rsa ... john
```

Since an SSH-based setup entails a fair amount of overhead, it may be more suited to temporary setups, such as for wireless VPNs. More permanent VPNs are better provided by tools such as **ipsecctl(8)** and **isakmpd(8)**.

ENVIRONMENT

ssh will normally set the following environment variables:

DISPLAY	The DISPLAY variable indicates the location of the X11 server. It is automatically set by ssh to point to a value of the form “hostname: <i>n</i> ”, where “hostname” indicates the host where the shell runs, and ‘ <i>n</i> ’ is an integer ≥ 1 . ssh uses this special value to forward X11 connections over the secure channel. The user should normally not set DISPLAY explicitly, as that will render the X11 connection insecure (and will require the user to manually copy any required authorization cookies).
HOME	Set to the path of the user’s home directory.
LOGNAME	Synonym for USER ; set for compatibility with systems that use this variable.
MAIL	Set to the path of the user’s mailbox.
PATH	Set to the default PATH , as specified when compiling ssh .
SSH_ASKPASS	If ssh needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If ssh does not have a terminal associated with it but DISPLAY and SSH_ASKPASS are set, it will execute the program specified by SSH_ASKPASS and open an X11 window to read the passphrase. This is particularly useful when calling ssh from a .xsession or related script. (Note that on some machines it may be neces-

	sary to redirect the input from <code>/dev/null</code> to make this work.)
<code>SSH_ASKPASS_REQUIRE</code>	Allows further control over the use of an askpass program. If this variable is set to “never” then ssh will never attempt to use one. If it is set to “prefer”, then ssh will prefer to use the askpass program instead of the TTY when requesting passwords. Finally, if the variable is set to “force”, then the askpass program will be used for all passphrase input regardless of whether <code>DISPLAY</code> is set.
<code>SSH_AUTH_SOCK</code>	Identifies the path of a UNIX-domain socket used to communicate with the agent.
<code>SSH_CONNECTION</code>	Identifies the client and server ends of the connection. The variable contains four space-separated values: client IP address, client port number, server IP address, and server port number.
<code>SSH_ORIGINAL_COMMAND</code>	This variable contains the original command line if a forced command is executed. It can be used to extract the original arguments.
<code>SSH_TTY</code>	This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.
<code>SSH_TUNNEL</code>	Optionally set by <code>sshd(8)</code> to contain the interface names assigned if tunnel forwarding was requested by the client.
<code>SSH_USER_AUTH</code>	Optionally set by <code>sshd(8)</code> , this variable may contain a pathname to a file that lists the authentication methods successfully used when the session was established, including any public keys that were used.
<code>TZ</code>	This variable is set to indicate the present time zone if it was set when the daemon was started (i.e. the daemon passes the value on to new connections).
<code>USER</code>	Set to the name of the user logging in.

Additionally, **ssh** reads `~/.ssh/environment`, and adds lines of the format “`VARNAME=value`” to the environment if the file exists and users are allowed to change their environment. For more information, see the **PermitUserEnvironment** option in `sshd_config(5)`.

FILES

`~/.rhosts`

This file is used for host-based authentication (see above). On some machines this file may need to be world-readable if the user’s home directory is on an NFS partition, because `sshd(8)` reads it as root. Additionally, this file must be owned by the user, and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user, and not accessible by others.

`~/.shosts`

This file is used in exactly the same way as `.rhosts`, but allows host-based authentication without permitting login with rlogin/rsh.

`~/.ssh/`

This directory is the default location for all user-specific configuration and authentication information. There is no general requirement to keep the entire contents of this directory secret, but the recommended permissions are read/write/execute for the user, and not accessible by others.

`~/.ssh/authorized_keys`

Lists the public keys (DSA, ECDSA, Ed25519, RSA) that can be used for logging in as this user. The format of this file is described in the `sshd(8)` manual page. This file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

`~/.ssh/config`

This is the per-user configuration file. The file format and configuration options are described in `ssh_config(5)`. Because of the potential for abuse, this file must have strict permissions: read/write for the user, and not writable by others. It may be group-writable provided that the group in question contains only the user.

`~/.ssh/environment`

Contains additional definitions for environment variables; see **ENVIRONMENT**, above.

`~/.ssh/id_dsa`

`~/.ssh/id_ecdsa`

`~/.ssh/id_ecdsa_sk`

`~/.ssh/id_ed25519`

`~/.ssh/id_ed25519_sk`

`~/.ssh/id_rsa`

Contains the private key for authentication. These files contain sensitive data and should be readable by the user but not accessible by others (read/write/execute). **ssh** will simply ignore a private key file if it is accessible by others. It is possible to specify a passphrase when generating the key which will be used to encrypt the sensitive part of this file using AES-128.

`~/.ssh/id_dsa.pub`

`~/.ssh/id_ecdsa.pub`

`~/.ssh/id_ecdsa_sk.pub`

`~/.ssh/id_ed25519.pub`

`~/.ssh/id_ed25519_sk.pub`

`~/.ssh/id_rsa.pub`

Contains the public key for authentication. These files are not sensitive and can (but need not) be readable by anyone.

`~/.ssh/known_hosts`

Contains a list of host keys for all hosts the user has logged into that are not already in the systemwide list of known host keys. See `sshd(8)` for further details of the format of this file.

`~/.ssh/rc`

Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. See the `sshd(8)` manual page for more information.

`/etc/hosts.equiv`

This file is for host-based authentication (see above). It should only be writable by root.

`/etc/ssh/shosts.equiv`

This file is used in exactly the same way as `hosts.equiv`, but allows host-based authentication without permitting login with rlogin/rsh.

`/etc/ssh/ssh_config`

Systemwide configuration file. The file format and configuration options are described in `ssh_config(5)`.

`/etc/ssh/ssh_host_key`

/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_ecdsa_key
/etc/ssh/ssh_host_ed25519_key
/etc/ssh/ssh_host_rsa_key

These files contain the private parts of the host keys and are used for host-based authentication.

/etc/ssh/ssh_known_hosts

Systemwide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. It should be world-readable. See sshd(8) for further details of the format of this file.

/etc/ssh/sshrc

Commands in this file are executed by **ssh** when the user logs in, just before the user's shell (or command) is started. See the sshd(8) manual page for more information.

EXIT STATUS

ssh exits with the exit status of the remote command or with 255 if an error occurred.

SEE ALSO

scp(1), sftp(1), ssh-add(1), ssh-agent(1), ssh-argv0(1), ssh-keygen(1), ssh-keyscan(1), tun(4), ssh_config(5), ssh-keysign(8), sshd(8)

STANDARDS

- S. Lehtinen and C. Lonwick, *The Secure Shell (SSH) Protocol Assigned Numbers*, RFC 4250, January 2006.
T. Ylonen and C. Lonwick, *The Secure Shell (SSH) Protocol Architecture*, RFC 4251, January 2006.
T. Ylonen and C. Lonwick, *The Secure Shell (SSH) Authentication Protocol*, RFC 4252, January 2006.
T. Ylonen and C. Lonwick, *The Secure Shell (SSH) Transport Layer Protocol*, RFC 4253, January 2006.
T. Ylonen and C. Lonwick, *The Secure Shell (SSH) Connection Protocol*, RFC 4254, January 2006.
J. Schlyter and W. Griffin, *Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints*, RFC 4255, January 2006.
F. Cusack and M. Forssen, *Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)*, RFC 4256, January 2006.
J. Galbraith and P. Remaker, *The Secure Shell (SSH) Session Channel Break Extension*, RFC 4335, January 2006.
M. Bellare, T. Kohno, and C. Namprempre, *The Secure Shell (SSH) Transport Layer Encryption Modes*, RFC 4344, January 2006.
B. Harris, *Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol*, RFC 4345, January 2006.
M. Friedl, N. Provos, and W. Simpson, *Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol*, RFC 4419, March 2006.
J. Galbraith and R. Thayer, *The Secure Shell (SSH) Public Key File Format*, RFC 4716, November 2006.
D. Stebila and J. Green, *Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer*, RFC 5656, December 2009.
A. Perrig and D. Song, *Hash Visualization: a New Technique to improve Real-World Security*, 1999, International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99).

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by [Tatu Ylonen](#). Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0.

NAME**sshd** — OpenSSH daemon**SYNOPSIS**

```
sshd [-46DdeiqTt] [-C connection_spec] [-c host_certificate_file]
      [-E log_file] [-f config_file] [-g login_grace_time]
      [-h host_key_file] [-o option] [-p port] [-u len]
```

DESCRIPTION

sshd (OpenSSH Daemon) is the daemon program for ssh(1). It provides secure encrypted communications between two untrusted hosts over an insecure network.

sshd listens for connections from clients. It is normally started at boot from /etc/init.d/ssh. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

sshd can be configured using command-line options or a configuration file (by default sshd_config(5)); command-line options override values specified in the configuration file. **sshd** rereads its configuration file when it receives a hangup signal, SIGHUP, by executing itself with the name and options it was started with, e.g. /usr/sbin/sshd.

The options are as follows:

-4 Forces **sshd** to use IPv4 addresses only.

-6 Forces **sshd** to use IPv6 addresses only.

-C *connection_spec*

Specify the connection parameters to use for the **-T** e xtended test mode. If provided, any **Match** directives in the configuration file that would apply are applied before the configuration is written to standard output. The connection parameters are supplied as keyword=value pairs and may be supplied in any order, either with multiple **-C** options or as a comma-separated list. The keywords are “addr”, “user”, “host”, “laddr”, “lport”, and “rdomain” and correspond to source address, user, resolved source host name, local address, local port number and routing domain respectively.

-c *host_certificate_file*

Specifies a path to a certificate file to identify **sshd** during key exchange. The certificate file must match a host key file specified using the **-h** option or the **HostKey** configuration directive.

-D When this option is specified, **sshd** will not detach and does not become a daemon. This allows easy monitoring of **sshd**.

-d Debug mode. The server sends verbose debug output to standard error, and does not put itself in the background. The server also will not fork(2) and will only process one connection. This option is only intended for debugging for the server. Multiple **-d** options increase the debugging level. Maximum is 3.

-E *log_file*

Append debug logs to *log_file* instead of the system log.

-e Write debug logs to standard error instead of the system log.

-f *config_file*

Specifies the name of the configuration file. The default is /etc/ssh/sshd_config. **sshd** refuses to start if there is no configuration file.

-g *login_grace_time*

Gives the grace time for clients to authenticate themselves (default 120 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. A value of zero indicates no limit.

-h *host_key_file*

Specifies a file from which a host key is read. This option must be given if **sshd** is not run as root (as the normal host key files are normally not readable by anyone but root). The default is `/etc/ssh/ssh_host_ecdsa_key`, `/etc/ssh/ssh_host_ed25519_key` and `/etc/ssh/ssh_host_rsa_key`. It is possible to have multiple host key files for the different host key algorithms.

-i Specifies that **sshd** is being run from **inetd(8)**.**-o** *option*

Can be used to give options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag. For full details of the options, and their values, see **sshd_config(5)**.

-p *port*

Specifies the port on which the server listens for connections (default 22). Multiple port options are permitted. Ports specified in the configuration file with the **Port** option are ignored when a command-line port is specified. Ports specified using the **ListenAddress** option override command-line ports.

-q Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.**-T** Extended test mode. Check the validity of the configuration file, output the effective configuration to stdout and then exit. Optionally, **Match** rules may be applied by specifying the connection parameters using one or more **-C** options.**-t** Test mode. Only check the validity of the configuration file and sanity of the keys. This is useful for updating **sshd** reliably as configuration options may change.**-u** *len*

This option is used to specify the size of the field in the **utmp** structure that holds the remote host name. If the resolved host name is longer than *len*, the dotted decimal value will be used instead. This allows hosts with very long host names that overflow this field to still be uniquely identified. Specifying **-u0** indicates that only dotted decimal addresses should be put into the **utmp** file. **-u0** may also be used to prevent **sshd** from making DNS requests unless the authentication mechanism or configuration requires it. Authentication mechanisms that may require DNS include **HostbasedAuthentication** and using a **from="pattern-list"** option in a key file. Configuration options that require DNS include using a **USER@HOST** pattern in **AllowUsers** or **DenyUsers**.

AUTHENTICATION

The OpenSSH SSH daemon supports SSH protocol 2 only. Each host has a host-specific key, used to identify the host. Whenever a client connects, the daemon responds with its public host key. The client compares the host key against its own database to verify that it has not changed. Forward secrecy is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (MAC).

Finally, the server and the client enter an authentication dialog. The client tries to authenticate itself using host-based authentication, public key authentication, challenge-response authentication, or password authentication.

Regardless of the authentication type, the account is checked to ensure that it is accessible. An account is not accessible if it is locked, listed in **DenyUsers** or its group is listed in **DenyGroups**. The definition of a locked account is system dependent. Some platforms have their own account database (eg AIX) and some modify the passwd field (*LK* on Solaris and UnixWare, '*' on HP-UX, containingNoLogin on Tru64, a leading *LOCKED* on FreeBSD and a leading '!' on most Linuxes). If there is a requirement to disable password authentication for the account while allowing still public-key, then the passwd field should be set to something other than these values (eg 'NP' or*NP*).

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time the client may request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP connections, or forwarding the authentication agent connection over the secure channel.

After this, the client either requests an interactive shell or execution or a non-interactive command, which **sshd** will execute via the user's shell using its **-c** option. The sides then enter session mode. In this mode, either side may send data at any time, and such data is forwarded to/from the shell or command on the server side, and the user terminal in the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

LOGIN PROCESS

When a user successfully logs in, **sshd** does the following:

1. If the login is on a tty, and no command has been specified, prints last login time and /etc/motd (unless prevented in the configuration file or by ~/.hushlogin; see the **FILES** section).
2. If the login is on a tty, records login time.
3. Checks /etc/nologin; if it exists, prints contents and quits (unless root).
4. Changes to run with normal user privileges.
5. Sets up basic environment.
6. Reads the file ~/.ssh/environment, if it exists, and users are allowed to change their environment. See the **PermitUserEnvironment** option in **sshd_config(5)**.
7. Changes to user's home directory.
8. If ~/.ssh/rc exists and the **sshd_config(5)** **PermitUserRC** option is set, runs it; else if /etc/ssh/sshrc exists, runs it; otherwise runs xauth(1). The "rc" files are given the X11 authentication protocol and cookie in standard input. See **SSHRC**, below.
9. Runs user's shell or command. All commands are run under the user's login shell as specified in the system password database.

SSHRC

If the file ~/.ssh/rc exists, sh(1) runs it after reading the environment files but before starting the user's shell or command. It must not produce any output on stdout; stderr must be used instead. If X11 forwarding is in use, it will receive the "proto cookie" pair in its standard input (and DISPLAY in its environment). The script must call xauth(1) because **sshd** will not run xauth automatically to add X11 cookies.

The primary purpose of this file is to run any initialization routines which may be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment.

This file will probably contain some initialization code followed by something similar to:

```
if read proto cookie && [ -n "$DISPLAY" ]; then
    if [ `echo $DISPLAY | cut -c1-10` = 'localhost:' ]; then
        # X11UseLocalhost=yes
        echo add unix:'echo $DISPLAY |
            cut -c11-' $proto $cookie
    else
        # X11UseLocalhost=no
        echo add $DISPLAY $proto $cookie
    fi | xauth -q -
fi
```

If this file does not exist, `/etc/ssh/sshrc` is run, and if that does not exist either, `xauth` is used to add the cookie.

AUTHORIZED_KEYS FILE FORMAT

AuthorizedKeysFile specifies the files containing public keys for public key authentication; if this option is not specified, the default is `~/.ssh/authorized_keys` and `~/.ssh/authorized_keys2`. Each line of the file contains one key (empty lines and lines starting with a '#' are ignored as comments). Public keys consist of the following space-separated fields: options, keytype, base64-encoded key, comment. The options field is optional. The supported key types are:

```
sk-ecdsa-sha2-nistp256@openssh.com
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
sk-ssh-ed25519@openssh.com
ssh-ed25519
ssh-dss
ssh-rsa
```

The comment field is not used for anything (but may be convenient for the user to identify the key).

Note that lines in this file can be several hundred bytes long (because of the size of the public key encoding) up to a limit of 8 kilobytes, which permits RSA keys up to 16 kilobits. You don't want to type them in; instead, copy the `id_dsa.pub`, `id_ecdsa.pub`, `id_ecdsa_sk.pub`, `id_ed25519.pub`, `id_ed25519_sk.pub`, or the `id_rsa.pub` file and edit it.

sshd enforces a minimum RSA key modulus size of 1024 bits.

The options (if present) consist of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported (note that option keywords are case-insensitive):

agent-forwarding

Enable authentication agent forwarding previously disabled by the **restrict** option.

cert-authority

Specifies that the listed key is a certification authority (CA) that is trusted to validate signed certificates for user authentication.

Certificates may encode access restrictions similar to these key options. If both certificate restrictions and key options are present, the most restrictive union of the two is applied.

command="command"

Specifies that the command is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a pty if the client requests a pty; otherwise it is run without a tty. If an 8-bit clean channel is required, one must not request a pty or should specify **no-pty**. A quote may be included in the command by quoting it with a backslash.

This option might be useful to restrict certain public keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Note that the client may specify TCP and/or X11 forwarding unless they are explicitly prohibited, e.g. using the **restrict** key option.

The command originally supplied by the client is available in the `SSH_ORIGINAL_COMMAND` environment variable. Note that this option applies to shell, command or subsystem execution. Also note that this command may be superseded by a `sshd_config(5)` **ForceCommand** directive.

If a command is specified and a forced-command is embedded in a certificate used for authentication, then the certificate will be accepted only if the two commands are identical.

environment="NAME=value"

Specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled via the **PermitUserEnvironment** option.

expiry-time="timespec"

Specifies a time after which the key will not be accepted. The time may be specified as a YYYYMMDDHHMM[SS] date or a YYYYMMDDHHMM[SS] time in the system time-zone.

from="pattern-list"

Specifies that in addition to public key authentication, either the canonical name of the remote host or its IP address must be present in the comma-separated list of patterns. See PATTERNS in `ssh_config(5)` for more information on patterns.

In addition to the wildcard matching that may be applied to hostnames or addresses, a **from** stanza may match IP addresses using CIDR address/masklen notation.

The purpose of this option is to optionally increase security: public key authentication by itself does not trust the network or name servers or anything (but the key); however, if somebody somehow steals the key, the key permits an intruder to log in from anywhere in the world. This additional option makes using a stolen key more difficult (name servers and/or routers would have to be compromised in addition to just the key).

no-agent-forwarding

Forbids authentication agent forwarding when this key is used for authentication.

no-port-forwarding

Forbids TCP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, e.g. in connection with the **command** option.

no-pty

Prevents tty allocation (a request to allocate a pty will fail).

no-user-rc

Disables execution of `~/.ssh/rc`.

no-X11-forwarding

Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

permitlisten="[:host:]port"

Limit remote port forwarding with the ssh(1) **-R** option such that it may only listen on the specified host (optional) and port. IPv6 addresses can be specified by enclosing the address in square brackets. Multiple **permitlisten** options may be applied separated by commas. Hostnames may include wildcards as described in the PATTERNS section in ssh_config(5). A port specification of * matches any port. Note that the setting of **GatewayPorts** may further restrict listen addresses. Note that ssh(1) will send a hostname of “localhost” if a listen host was not specified when the forwarding was requested, and that this name is treated differently to the explicit localhost addresses “127.0.0.1” and “::1”.

permitopen="host:port"

Limit local port forwarding with the ssh(1) **-L** option such that it may only connect to the specified host and port. IPv6 addresses can be specified by enclosing the address in square brackets. Multiple **permitopen** options may be applied separated by commas. No pattern matching or name lookup is performed on the specified hostnames, they must be literal host names and/or addresses. A port specification of * matches any port.

port-forwarding

Enable port forwarding previously disabled by the **restrict** option.

principals="principals"

On a **cert-authority** line, specifies allowed principals for certificate authentication as a comma-separated list. At least one name from the list must appear in the certificate’s list of principals for the certificate to be accepted. This option is ignored for keys that are not marked as trusted certificate signers using the **cert-authority** option.

pty

Permits tty allocation previously disabled by the **restrict** option.

no-touch-required

Do not require demonstration of user presence for signatures made using this key. This option only makes sense for the FIDO authenticator algorithms **ecdsa-sk** and **ed25519-sk**.

verify-required

Require that signatures made using this key attest that they verified the user, e.g. via a PIN. This option only makes sense for the FIDO authenticator algorithms **ecdsa-sk** and **ed25519-sk**.

restrict

Enable all restrictions, i.e. disable port, agent and X11 forwarding, as well as disabling PTY allocation and execution of ~/.ssh/rc. If any future restriction capabilities are added to authorized_keys files they will be included in this set.

tunnel="n"

Force a tun(4) device on the server. Without this option, the next available device will be used if the client requests a tunnel.

user-rc

Enables execution of ~/.ssh/rc previously disabled by the **restrict** option.

x11-forwarding

Permits X11 forwarding previously disabled by the **restrict** option.

An example authorized_keys file:

```
# Comments are allowed at start of line. Blank lines are allowed.
# Plain key, no restrictions
ssh-rsa ...
# Forced command, disable PTY and all forwarding
restrict,command="dump /home" ssh-rsa ...
```

```

# Restriction of ssh -L forwarding destinations
permitopen="192.0.2.1:80",permitopen="192.0.2.2:25" ssh-rsa ...
# Restriction of ssh -R forwarding listeners
permitlisten="localhost:8080",permitlisten="[:1]:22000" ssh-rsa ...
# Configuration for tunnel forwarding
tunnel="0",command="sh /etc/netstart tun0" ssh-rsa ...
# Override of restriction to allow PTY allocation
restrict,pty,command="nethack" ssh-rsa ...
# Allow FIDO key without requiring touch
no-touch-required sk-ecdsa-sha2-nistp256@openssh.com ...
# Require user-verification (e.g. PIN or biometric) for FIDO key
verify-required sk-ecdsa-sha2-nistp256@openssh.com ...
# Trust CA key, allow touch-less FIDO if requested in certificate
cert-authority,no-touch-required,principals="user_a" ssh-rsa ...

```

SSH_KNOWN_HOSTS FILE FORMAT

The `/etc/ssh/ssh_known_hosts` and `~/.ssh/known_hosts` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically: whenever the user connects to an unknown host, its key is added to the per-user file.

Each line in these files contains the following fields: marker (optional), hostnames, keytype, base64-encoded key, comment. The fields are separated by spaces.

The marker is optional, but if it is present then it must be one of “`@cert-authority`”, to indicate that the line contains a certification authority (CA) key, or “`@revoked`”, to indicate that the key contained on the line is revoked and must not ever be accepted. Only one marker should be used on a key line.

Hostnames is a comma-separated list of patterns (“`*`” and “`?`” act as wildcards); each pattern in turn is matched against the host name. When `sshd` is authenticating a client, such as when using **HostbasedAuthentication**, this will be the canonical client host name. When `ssh(1)` is authenticating a server, this will be the host name given by the user, the value of the `ssh(1)` **HostkeyAlias** if it was specified, or the canonical server hostname if the `ssh(1)` **CanonicalizeHostname** option was used.

A pattern may also be preceded by ‘`!`’ to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line. A hostname or address may optionally be enclosed within ‘`[`’ and ‘`]`’ brackets then followed by ‘`:`’ and a non-standard port number.

Alternately, hostnames may be stored in a hashed form which hides host names and addresses should the file’s contents be disclosed. Hashed hostnames start with a ‘`]`’ character. Only one hashed hostname may appear on a single line and none of the above negation or wildcard operators may be applied.

The keytype and base64-encoded key are taken directly from the host key; they can be obtained, for example, from `/etc/ssh/ssh_host_rsa_key.pub`. The optional comment field continues to the end of the line, and is not used.

Lines starting with ‘`#`’ and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key; either one that matches exactly or, if the server has presented a certificate for authentication, the key of the certification authority that signed the certificate. For a key to be trusted as a certification authority, it must use the “`@cert-authority`” marker described above.

The known hosts file also provides a facility to mark keys as revoked, for example when it is known that the associated private key has been stolen. Revoked keys are specified by including the “`@revoked`” marker at the beginning of the key line, and are never accepted for authentication or as certification authorities, but in-

stead will produce a warning from `ssh(1)` when they are encountered.

It is permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

Note that the lines in these files are typically hundreds of characters long, and you definitely don't want to type in the host keys by hand. Rather, generate them by a script, `ssh-keyscan(1)` or by taking, for example, `/etc/ssh/ssh_host_rsa_key.pub` and adding the host names at the front. `ssh-keygen(1)` also offers some basic automated editing for `~/.ssh/known_hosts` including removing hosts matching a host name and converting all host names to their hashed representations.

An example `ssh_known_hosts` file:

```
# Comments allowed at start of line
closenet,...,192.0.2.53 1024 37 159...93 closenet.example.net
cvs.example.net,192.0.2.10 ssh-rsa AAAA1234....=
# A hashed hostname
|1|JfKTdBh7rNbXkVAQCRp4OQoPfmI=|USECr3SWf1JUPsms5Aqfd5QfxkM= ssh-rsa
AAAAA1234....=
# A revoked key
@revoked * ssh-rsa AAAAB5W...
# A CA key, accepted for any host in *.mydomain.com or *.mydomain.org
@cert-authority *.mydomain.org,*.mydomain.com ssh-rsa AAAAB5W...
```

FILES

`~/.hushlogin`

This file is used to suppress printing the last login time and `/etc/motd`, if **PrintLastLog** and **PrintMotd**, respectively, are enabled. It does not suppress printing of the banner specified by **Banner**.

`~/.rhosts`

This file is used for host-based authentication (see `ssh(1)` for more information). On some machines this file may need to be world-readable if the user's home directory is on an NFS partition, because `sshd` reads it as root. Additionally, this file must be owned by the user, and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user, and not accessible by others.

`~/.shosts`

This file is used in exactly the same way as `.rhosts`, but allows host-based authentication without permitting login with `rlogin/rsh`.

`~/.ssh`

This directory is the default location for all user-specific configuration and authentication information. There is no general requirement to keep the entire contents of this directory secret, but the recommended permissions are read/write/execute for the user, and not accessible by others.

`~/.ssh/authorized_keys`

Lists the public keys (DSA, ECDSA, Ed25519, RSA) that can be used for logging in as this user. The format of this file is described above. The content of the file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

If this file, the `~/.ssh` directory, or the user's home directory are writable by other users, then the file could be modified or replaced by unauthorized users. In this case, `sshd` will not allow it to be used unless the **StrictModes** option has been set to "no".

`~/.ssh/environment`

This file is read into the environment at login (if it exists). It can only contain empty lines, comment lines (that start with '#'), and assignment lines of the form name=value. The file should be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled via the **PermitUserEnvironment** option.

`~/.ssh/known_hosts`

Contains a list of host keys for all hosts the user has logged into that are not already in the systemwide list of known host keys. The format of this file is described above. This file should be writable only by root/the owner and can, but need not be, world-readable.

`~/.ssh/rc`

Contains initialization routines to be run before the user's home directory becomes accessible. This file should be writable only by the user, and need not be readable by anyone else.

`/etc/hosts.allow`

`/etc/hosts.deny`

Access controls that should be enforced by tcp-wrappers are defined here. Further details are described in `hosts_access(5)`.

`/etc/hosts.equiv`

This file is for host-based authentication (see `ssh(1)`). It should only be writable by root.

`/etc/ssh/moduli`

Contains Diffie-Hellman groups used for the "Diffie-Hellman Group Exchange" key exchange method. The file format is described in `moduli(5)`. If no usable groups are found in this file then fixed internal groups will be used.

`/etc/motd`

See `motd(5)`.

`/etc/nologin`

If this file exists, **sshd** refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in, and non-root connections are refused. The file should be world-readable.

`/etc/ssh/shosts.equiv`

This file is used in exactly the same way as `hosts.equiv`, but allows host-based authentication without permitting login with rlogin/rsh.

`/etc/ssh/ssh_host_ecdsa_key`

`/etc/ssh/ssh_host_ed25519_key`

`/etc/ssh/ssh_host_rsa_key`

These files contain the private parts of the host keys. These files should only be owned by root, readable only by root, and not accessible to others. Note that **sshd** does not start if these files are group/world-accessible.

`/etc/ssh/ssh_host_ecdsa_key.pub`

`/etc/ssh/ssh_host_ed25519_key.pub`

`/etc/ssh/ssh_host_rsa_key.pub`

These files contain the public parts of the host keys. These files should be world-readable but writable only by root. Their contents should match the respective private parts. These files are not really used for anything; they are provided for the convenience of the user so their contents can be copied to known hosts files. These files are created using `ssh-keygen(1)`.

/etc/ssh/ssh_known_hosts

Systemwide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. The format of this file is described above. This file should be writable only by root/the owner and should be world-readable.

/etc/ssh/sshd_config

Contains configuration data for **sshd**. The file format and configuration options are described in **sshd_config(5)**.

/etc/ssh/sshrc

Similar to `~/.ssh/rc`, it can be used to specify machine-specific login-time initializations globally. This file should be writable only by root, and should be world-readable.

/run/sshd

`chroot(2)` directory used by **sshd** during privilege separation in the pre-authentication phase. The directory should not contain any files and must be owned by root and not group or world-writable.

/run/sshd.pid

Contains the process ID of the **sshd** listening for connections (if there are several daemons running concurrently for different ports, this contains the process ID of the one started last). The content of this file is not sensitive; it can be world-readable.

SEE ALSO

scp(1), sftp(1), ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), ssh-keyscan(1), chroot(2), hosts_access(5), moduli(5), sshd_config(5), inetd(8), sftp-server(8)

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, re-added newer features and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0. Niels Provos and Markus Friedl contributed support for privilege separation.

NAME

sudo, sudoedit — execute a command as another user

SYNOPSIS

```
sudo -h | -K | -k | -v
sudo -v[ -ABknS] [-g group] [-h host] [-p prompt] [-u user]
sudo -l[ -ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
sudo [ -ABbEHnPS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
      [-R directory] [-r role] [-t type] [-T timeout] [-u user] [VAR=value]
      [-i | -s] [command]
sudoedit [ -ABknS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
      [-R directory] [-r role] [-t type] [-T timeout] [-u user] file ...
```

DESCRIPTION

sudo allows a permitted user to execute a *command* as the superuser or another user, as specified by the security policy. The invoking user's real (*not effective*) user-ID is used to determine the user name with which to query the security policy.

sudo supports a plugin architecture for security policies, auditing, and input/output logging. Third parties can develop and distribute their own plugins to work seamlessly with the **sudo** front-end. The default security policy is *sudoers*, which is configured via the file /etc/sudoers, or via LDAP. See the **Plugins** section for more information.

The security policy determines what privileges, if any, a user has to run **sudo**. The policy may require that users authenticate themselves with a password or another authentication mechanism. If authentication is required, **sudo** will exit if the user's password is not entered within a configurable time limit. This limit is policy-specific; the default password prompt timeout for the *sudoers* security policy is 0 minutes.

Security policies may support credential caching to allow the user to run **sudo** again for a period of time without requiring authentication. By default, the *sudoers* policy caches credentials on a per-terminal basis for 15 minutes. See the *timestamp_type* and *timestamp_timeout* options in *sudoers*(5) for more information. By running **sudo** with the **-v** option, a user can update the cached credentials without running a *command*.

On systems where **sudo** is the primary method of gaining superuser privileges, it is imperative to avoid syntax errors in the security policy configuration files. For the default security policy, *sudoers*(5), changes to the configuration files should be made using the *visudo*(8) utility which will ensure that no syntax errors are introduced.

When invoked as **sudoedit**, the **-e** option (described below), is implied.

Security policies and audit plugins may log successful and failed attempts to run **sudo**. If an I/O plugin is configured, the running command's input and output may be logged as well.

The options are as follows:

-A, --askpass

Normally, if **sudo** requires a password, it will read it from the user's terminal. If the **-A** (*askpass*) option is specified, a (possibly graphical) helper program is executed to read the user's password and output the password to the standard output. If the **SUDO_ASKPASS** environment variable is set, it specifies the path to the helper program. Otherwise, if *sudo.conf*(5) contains a line specifying the *askpass* program, that value will be used. For example:

```
# Path to askpass helper program
Path askpass /usr/X11R6/bin/ssh-askpass
```

If no askpass program is available, **sudo** will exit with an error.

-B, --bell

Ring the bell as part of the password prompt when a terminal is present. This option has no effect if an askpass program is used.

-b, --background

Run the given command in the background. Note that it is not possible to use shell job control to manipulate background processes started by **sudo**. Most interactive commands will fail to work properly in background mode.

-C num, --close-from=num

Close all file descriptors greater than or equal to *num* before executing a command. Values less than three are not permitted. By default, **sudo** will close all open file descriptors other than standard input, standard output, and standard error when executing a command. The security policy may restrict the user's ability to use this option. The *sudoers* policy only permits use of the **-C** option when the administrator has enabled the *closefrom_override* option.

-D directory, --chdir=directory

Run the command in the specified *directory* instead of the current working directory. The security policy may return an error if the user does not have permission to specify the working directory.

-E, --preserve-env

Indicates to the security policy that the user wishes to preserve their existing environment variables. The security policy may return an error if the user does not have permission to preserve the environment.

--preserve-env=list

Indicates to the security policy that the user wishes to add the comma-separated list of environment variables to those preserved from the user's environment. The security policy may return an error if the user does not have permission to preserve the environment. This option may be specified multiple times.

-e, --edit

Edit one or more files instead of running a command. In lieu of a path name, the string "sudoeedit" is used when consulting the security policy. If the user is authorized by the policy, the following steps are taken:

1. Temporary copies are made of the files to be edited with the owner set to the invoking user.
2. The editor specified by the policy is run to edit the temporary files. The *sudoers* policy uses the *SUDO_EDITOR*, *VISUAL* and *EDITOR* environment variables (in that order). If none of *SUDO_EDITOR*, *VISUAL* or *EDITOR* are set, the first program listed in the *editor sudoers(5)* option is used.
3. If they have been modified, the temporary files are copied back to their original location and the temporary versions are removed.

To help prevent the editing of unauthorized files, the following restrictions are enforced unless explicitly allowed by the security policy:

- Symbolic links may not be edited (version 1.8.15 and higher).
- Symbolic links along the path to be edited are not followed when the parent directory is writable by the invoking user unless that user is root (version 1.8.16 and higher).

- Files located in a directory that is writable by the invoking user may not be edited unless that user is root (version 1.8.16 and higher).

Users are never allowed to edit device special files.

If the specified file does not exist, it will be created. Note that unlike most commands run by **sudo**, the editor is run with the invoking user's environment unmodified. If the temporary file becomes empty after editing, the user will be prompted before it is installed. If, for some reason, **sudo** is unable to update a file with its edited version, the user will receive a warning and the edited copy will remain in a temporary file.

-g group, --group=group

Run the command with the primary group set to *group* instead of the primary group specified by the target user's password database entry. The *group* may be either a group name or a numeric group-ID (GID) prefixed with the '#' character (e.g., #0 for GID 0). When running a command as a GID, many shells require that the '#' be escaped with a backslash ('\'). If no **-u** option is specified, the command will be run as the invoking user. In either case, the primary group will be set to *group*. The *sudoer*'s policy permits any of the target user's groups to be specified via the **-g** option as long as the **-P** option is not in use.

-H, --set-home

Request that the security policy set the HOME environment variable to the home directory specified by the target user's password database entry. Depending on the policy, this may be the default behavior.

-h, --help

Display a short help message to the standard output and exit.

-h host, --host=host

Run the command on the specified *host* if the security policy plugin supports remote commands. Note that the *sudoer*'s plugin does not currently support running remote commands. This may also be used in conjunction with the **-l** option to list a user's privileges for the remote host.

-i, --login

Run the shell specified by the target user's password database entry as a login shell. This means that login-specific resource files such as *.profile*, *.bash_profile*, or *.login* will be read by the shell. If a command is specified, it is passed to the shell as a simple command using the **-c** option. The command and any arguments are concatenated, separated by spaces, after escaping each character (including white space) with a backslash ('\') except for alphanumerics, underscores, hyphens, and dollar signs. If no command is specified, an interactive shell is executed. **sudo** attempts to change to that user's home directory before running the shell. The command is run with an environment similar to the one a user would receive at log in. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details. The *Command environment* section in the *sudoers(5)* manual documents how the **-i** option affects the environment in which a command is run when the *sudoers* policy is in use.

-K, --remove-timestamp

Similar to the **-k** option, except that it removes the user's cached credentials entirely and may not be used in conjunction with a command or other option. This option does not require a password. Not all security policies support credential caching.

-k, --reset-timestamp

When used without a command, invalidates the user's cached credentials. In other words, the next time **sudo** is run a password will be required. This option does not require a password,

and was added to allow a user to revoke **sudo** permissions from a .logout file.

When used in conjunction with a command or an option that may require a password, this option will cause **sudo** to ignore the user's cached credentials. As a result, **sudo** will prompt for a password (if one is required by the security policy) and will not update the user's cached credentials.

Not all security policies support credential caching.

-l, --list

If no *command* is specified, list the allowed (and forbidden) commands for the invoking user (or the user specified by the **-U** option) on the current host. A longer list format is used if this option is specified multiple times and the security policy supports a verbose output format.

If a *command* is specified and is permitted by the security policy, the fully-qualified path to the command is displayed along with any command line arguments. If a *command* is specified but not allowed by the policy, **sudo** will exit with a status value of 1.

-n, --non-interactive

Avoid prompting the user for input of any kind. If a password is required for the command to run, **sudo** will display an error message and exit.

-P, --preserve-groups

Preserve the invoking user's group vector unaltered. By default, the *sudoers* policy will initialize the group vector to the list of groups the target user is a member of. The real and effective group-IDs, however, are still set to match the target user.

-p *prompt*, --prompt=*prompt*

Use a custom password prompt with optional escape sequences. The following percent ('%') escape sequences are supported by the *sudoers* policy:

%H

expanded to the host name including the domain name (only if the machine's host name is fully qualified or the *fqdn* option is set in *sudoers*(5))

%h

expanded to the local host name without the domain name

%p

expanded to the name of the user whose password is being requested (respects the *rootpw*, *targetpw*, and *runaspw* flags in *sudoers*(5))

%U

expanded to the login name of the user the command will be run as (defaults to root unless the **-u** option is also specified)

%u

expanded to the invoking user's login name

%%

two consecutive '%' characters are collapsed into a single '%' character

The custom prompt will override the default prompt specified by either the security policy or the *SUDO_PROMPT* environment variable. On systems that use PAM, the custom prompt will also override the prompt specified by a PAM module unless the *passprompt_override* flag is disabled in *sudoers*.

-R directory, --chroot=directory

Change to the specified root *directory* (see *chroot(8)*) before running the command. The security policy may return an error if the user does not have permission to specify the root directory.

-r role, --role=role

Run the command with an SELinux security context that includes the specified *role*.

-s, --stdin

Write the prompt to the standard error and read the password from the standard input instead of using the terminal device.

-s, --shell

Run the shell specified by the SHELL environment variable if it is set or the shell specified by the invoking user's password database entry. If a command is specified, it is passed to the shell as a simple command using the **-c** option. The command and any arguments are concatenated, separated by spaces, after escaping each character (including white space) with a backslash ('\\') except for alphanumerics, underscores, hyphens, and dollar signs. If no command is specified, an interactive shell is executed. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details.

-t type, --type=type

Run the command with an SELinux security context that includes the specified *type*. If no *type* is specified, the default type is derived from the role.

-U user, --other-user=user

Used in conjunction with the **-l** option to list the privileges for *user* instead of for the invoking user. The security policy may restrict listing other users' privileges. The *sudoer*'s policy only allows root or a user with the ALL privilege on the current host to use this option.

-T timeout, --command-timeout=timeout

Used to set a timeout for the command. If the timeout expires before the command has exited, the command will be terminated. The security policy may restrict the ability to set command timeouts. The *sudoer*'s policy requires that user-specified timeouts be explicitly enabled.

-u user, --user=user

Run the command as a user other than the default target user (usually *root*). The *user* may be either a user name or a numeric user-ID (UID) prefixed with the '#' character (e.g.,#0 for UID 0). When running commands as a UID, many shells require that the '#' be escaped with a backslash ('\\'). Some security policies may restrict UIDs to those listed in the password database. The *sudoer*'s policy allows UIDs that are not in the password database as long as the *targetpw* option is not set. Other security policies may not support this.

-v, --version

Print the **sudo** version string as well as the version string of any configured plugins. If the invoking user is already root, the **-V** option will display the arguments passed to configure when **sudo** was built; plugins may display additional information such as default options.

-v, --validate

Update the user's cached credentials, authenticating the user if necessary. For the *sudoers* plugin, this extends the **sudo** timeout for another 15 minutes by default, but does not run a command. Not all security policies support cached credentials.

-- The **--** option indicates that **sudo** should stop processing command line arguments.

Options that take a value may only be specified once unless otherwise indicated in the description. This is to help guard against problems caused by poorly written scripts that invoke **sudo** with user-controlled input.

Environment variables to be set for the command may also be passed on the command line in the form of `VAR=value`, e.g., `LD_LIBRARY_PATH=/usr/local/pkg/lib`. Variables passed on the command line are subject to restrictions imposed by the security policy plugin. The *sudoers* policy subjects variables passed on the command line to the same restrictions as normal environment variables with one important exception. If the `setenv v` option is set in *sudoers*, the command to be run has the `SETENV` tag set or the command matched is `ALL`, the user may set variables that would otherwise be forbidden. See *sudoers*(5) for more information.

COMMAND EXECUTION

When **sudo** executes a command, the security policy specifies the execution environment for the command. Typically, the real and effective user and group and IDs are set to match those of the target user, as specified in the password database, and the group vector is initialized based on the group database (unless the `-P` option was specified).

The following parameters may be specified by security policy:

- real and effective user-ID
- real and effective group-ID
- supplementary group-IDs
- the environment list
- current working directory
- file creation mode mask (umask)
- SELinux role and type
- scheduling priority (aka nice value)

Process model

There are two distinct ways **sudo** can run a command.

If an I/O logging plugin is configured or if the security policy explicitly requests it, a new pseudo-terminal (“pty”) is allocated and `fork(2)` is used to create a second **sudo** process, referred to as the *monitor*. The *monitor* creates a new terminal session with itself as the leader and the pty as its controlling terminal, calls `fork(2)`, sets up the execution environment as described above, and then uses the `execve(2)` system call to run the command in the child process. The *monitor* exists to relay job control signals between the user’s existing terminal and the pty the command is being run in. This makes it possible to suspend and resume the command. Without the monitor, the command would be in what POSIX terms an “orphaned process group” and it would not receive any job control signals from the kernel. When the command exits or is terminated by a signal, the *monitor* passes the command’s exit status to the main **sudo** process and exits. After receiving the command’s exit status, the main **sudo** passes the command’s exit status to the security policy’s close function and exits.

If no pty is used, **sudo** calls `fork(2)`, sets up the execution environment as described above, and uses the `execve(2)` system call to run the command in the child process. The main **sudo** process waits until the command has completed, then passes the command’s exit status to the security policy’s close function and exits. As a special case, if the policy plugin does not define a close function, **sudo** will execute the command directly instead of calling `fork(2)` first. The *sudoers* policy plugin will only define a close function when I/O logging is enabled, a pty is required, an SELinux role is specified, the command has an associated timeout, or the *pam_session* or *pam_setcred* options are enabled. Note that *pam_session* and *pam_setcred* are enabled by default on systems using PAM.

On systems that use PAM, the security policy's close function is responsible for closing the PAM session. It may also log the command's exit status.

Signal handling

When the command is run as a child of the **sudo** process, **sudo** will relay signals it receives to the command. The SIGINT and SIGQUIT signals are only relayed when the command is being run in a new pty or when the signal was sent by a user process, not the kernel. This prevents the command from receiving SIGINT twice each time the user enters control-C. Some signals, such as SIGSTOP and SIGKILL, cannot be caught and thus will not be relayed to the command. As a general rule, SIGTSTP should be used instead of SIGSTOP when you wish to suspend a command being run by **sudo**.

As a special case, **sudo** will not relay signals that were sent by the command it is running. This prevents the command from accidentally killing itself. On some systems, the **reboot(8)** command sends SIGTERM to all non-system processes other than itself before rebooting the system. This prevents **sudo** from relaying the SIGTERM signal it received back to **reboot(8)**, which might then exit before the system was actually rebooted, leaving it in a half-dead state similar to single user mode. Note, however, that this check only applies to the command run by **sudo** and not any other processes that the command may create. As a result, running a script that calls **reboot(8)** or **shutdown(8)** via **sudo** may cause the system to end up in this undefined state unless the **reboot(8)** or **shutdown(8)** are run using the **exec()** family of functions instead of **system()** (which interposes a shell between the command and the calling process).

If no I/O logging plugins are loaded and the policy plugin has not defined a **close()** function, set a command timeout, or required that the command be run in a new pty, **sudo** may execute the command directly instead of running it as a child process.

Plugins

Plugins may be specified via Plugin directives in the **sudo.conf(5)** file. They may be loaded as dynamic shared objects (on systems that support them), or compiled directly into the **sudo** binary. If no **sudo.conf(5)** file is present, or if it doesn't contain any Plugin lines, **sudo** will use **sudoers(5)** for the policy, auditing, and I/O logging plugins. See the **sudo.conf(5)** manual for details of the **/etc/sudo.conf** file and the **sudo_plugin(5)** manual for more information about the **sudo** plugin architecture.

EXIT VALUE

Upon successful execution of a command, the exit status from **sudo** will be the exit status of the program that was executed. If the command terminated due to receipt of a signal, **sudo** will send itself the same signal that terminated the command.

If the **-l** option was specified without a command, **sudo** will exit with a value of 0 if the user is allowed to run **sudo** and they authenticated successfully (as required by the security policy). If a command is specified with the **-l** option, the exit value will only be 0 if the command is permitted by the security policy, otherwise it will be 1.

If there is an authentication failure, a configuration/permission problem, or if the given command cannot be executed, **sudo** exits with a value of 1. In the latter case, the error string is printed to the standard error. If **sudo** cannot stat(2) one or more entries in the user's PATH, an error is printed to the standard error. (If the directory does not exist or if it is not really a directory, the entry is ignored and no error is printed.) This should not happen under normal circumstances. The most common reason for stat(2) to return "permission denied" is if you are running an automounter and one of the directories in your PATH is on a machine that is currently unreachable.

SECURITY NOTES

sudo tries to be safe when executing external commands.

To prevent command spoofing, **sudo** checks "." and "" (both denoting current directory) last when searching for a command in the user's PATH (if one or both are in the PATH). Depending on the security policy, the user's PATH environment variable may be modified, replaced, or passed unchanged to the program that **sudo** executes.

Users should *never* be granted **sudo** privileges to execute files that are writable by the user or that reside in a directory that is writable by the user. If the user can modify or replace the command there is no way to limit what additional commands they can run.

Please note that **sudo** will normally only log the command it explicitly runs. If a user runs a command such as **sudo su** or **sudo sh**, subsequent commands run from that shell are not subject to **sudo**'s security policy. The same is true for commands that offer shell escapes (including most editors). If I/O logging is enabled, subsequent commands will have their input and/or output logged, but there will not be traditional logs for those commands. Because of this, care must be taken when giving users access to commands via **sudo** to verify that the command does not inadvertently give the user an effective root shell. For information on ways to address this, please see the *Preventing shell escapes* section in **sudoers(5)**.

To prevent the disclosure of potentially sensitive information, **sudo** disables core dumps by default while it is executing (they are re-enabled for the command that is run). This historical practice dates from a time when most operating systems allowed set-user-ID processes to dump core by default. To aid in debugging **sudo** crashes, you may wish to re-enable core dumps by setting "disable_coredump" to false in the **sudo.conf(5)** file as follows:

```
Set disable_coredump false
```

See the **sudo.conf(5)** manual for more information.

ENVIRONMENT

sudo utilizes the following environment variables. The security policy has control over the actual content of the command's environment.

EDITOR	Default editor to use in -e (sudoedit) mode if neither SUDO_EDITOR nor VISUAL is set.
MAIL	Set to the mail spool of the target user when the -i option is specified, or when env_reset is enabled in sudoers (unless MAIL is present in the env_keep list).
HOME	Set to the home directory of the target user when the -i or -H options are specified, when the -s option is specified and set_home is set in sudoers , when always_set_home is enabled in sudoers , or when env_reset is enabled in sudoers and HOME is not present in the env_keep list.
LOGNAME	Set to the login name of the target user when the -i option is specified, when the set_logname option is enabled in sudoers , or when the env_reset option is enabled in sudoers (unless LOGNAME is present in the env_keep list).
PATH	May be overridden by the security policy.
SHELL	Used to determine shell to run with -s option.
SUDO_ASKPASS	Specifies the path to a helper program used to read the password if no terminal is available or if the -A option is specified.
SUDO_COMMAND	Set to the command run by sudo, including command line arguments. The command line arguments are truncated at 4096 characters to prevent a potential execution error.

SUDO_EDITOR	Default editor to use in -e (sudoedit) mode.
SUDO_GID	Set to the group-ID of the user who invoked sudo.
SUDO_PROMPT	Used as the default password prompt unless the -p option was specified.
SUDO_PS1	If set, PS1 will be set to its value for the program being run.
SUDO_UID	Set to the user-ID of the user who invoked sudo.
SUDO_USER	Set to the login name of the user who invoked sudo.
USER	Set to the same value as LOGNAME, described above.
VISUAL	Default editor to use in -e (sudoedit) mode if SUDO_EDITOR is not set.

FILES

/etc/sudo.conf	sudo front-end configuration
----------------	-------------------------------------

EXAMPLES

Note: the following examples assume a properly configured security policy.

To get a file listing of an unreadable directory:

```
$ sudo ls /usr/local/protected
```

To list the home directory of user yaz on a machine where the file system holding ~yaz is not exported as root:

```
$ sudo -u yaz ls ~yaz
```

To edit the index.html file as user www:

```
$ sudoedit -u www ~www/htdocs/index.html
```

To view system logs only accessible to root and users in the adm group:

```
$ sudo -g adm more /var/log/syslog
```

To run an editor as jim with a different primary group:

```
$ sudoedit -u jim -g audio ~jim/sound.txt
```

To shut down a machine:

```
$ sudo shutdown -r +15 "quick reboot"
```

To make a usage listing of the directories in the /home partition. Note that this runs the commands in a sub-shell to make the cd and file redirection work.

```
$ sudo sh -c "cd /home ; du -s * | sort -rn > USAGE"
```

DIAGNOSTICS

Error messages produced by **sudo** include:

editing files in a writable directory is not permitted

By default, **sudoedit** does not permit editing a file when any of the parent directories are writable by the invoking user. This avoids a race condition that could allow the user to overwrite an arbitrary file. See the *sudoedit_c heckdir* option in sudoers(5) for more information.

editing symbolic links is not permitted

By default, **sudoedit** does not follow symbolic links when opening files. See the *sudoedit_follow* option in sudoers(5) for more information.

effective uid is not 0, is sudo installed setuid root?

sudo was not run with root privileges. The **sudo** binary must be owned by the root user and have the set-user-ID bit set. Also, it must not be located on a file system mounted with the ‘nosuid’ option or on an NFS file system that maps uid 0 to an unprivileged uid.

effective uid is not 0, is sudo on a file system with the ‘nosuid’ option set or an NFS file system without root privileges?

sudo was not run with root privileges. The **sudo** binary has the proper owner and permissions but it still did not run with root privileges. The most common reason for this is that the file system the **sudo** binary is located on is mounted with the ‘nosuid’ option or it is an NFS file system that maps uid 0 to an unprivileged uid.

fatal error, unable to load plugins

An error occurred while loading or initializing the plugins specified in **sudo.conf(5)**.

invalid environment variable name

One or more environment variable names specified via the **-E** option contained an equal sign (‘=’). The arguments to the **-E** option should be environment variable names without an associated value.

no password was provided

When **sudo** tried to read the password, it did not receive any characters. This may happen if no terminal is available (or the **-s** option is specified) and the standard input has been redirected from **/dev/null**.

a terminal is required to read the password

sudo needs to read the password but there is no mechanism available for it to do so. A terminal is not present to read the password from, **sudo** has not been configured to read from the standard input, the **-s** option was not used, and no askpass helper has been specified either via the **sudo.conf(5)** file or the **SUDO_ASKPASS** environment variable.

no writable temporary directory found

sudoedit was unable to find a usable temporary directory in which to store its intermediate files.

The “no new privileges” flag is set, which prevents sudo from running as root.

sudo was run by a process that has the Linux “no new privileges” flag is set. This causes the set-user-ID bit to be ignored when running an executable, which will prevent **sudo** from functioning. The most likely cause for this is running **sudo** within a container that sets this flag. Check the documentation to see if it is possible to configure the container such that the flag is not set.

sudo must be owned by uid 0 and have the setuid bit set

sudo was not run with root privileges. The **sudo** binary does not have the correct owner or permissions. It must be owned by the root user and have the set-user-ID bit set.

sudoedit is not supported on this platform

It is only possible to run **sudoedit** on systems that support setting the effective user-ID.

timed out reading password

The user did not enter a password before the password timeout (5 minutes by default) expired.

you do not exist in the passwd database

Your user-ID does not appear in the system passwd database.

you may not specify environment variables in edit mode

It is only possible to specify environment variables when running a command. When editing a file, the editor is run with the user’s environment unmodified.

SEE ALSO

`su(1)`, `stat(2)`, `login_cap(3)`, `passwd(5)`, `sudo.conf(5)`, `sudo_plugin(5)`, `sudoers(5)`,
`sudoers_timestamp(5)`, `sudoreplay(8)`, `visudo(8)`

HISTORY

See the HISTORY file in the `sudo` distribution (<https://www.sudo.ws/history.html>) for a brief history of `sudo`.

AUTHORS

Many people have worked on `sudo` over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the `sudo` distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to `sudo`.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if that user is allowed to run arbitrary commands via `sudo`. Also, many programs (such as editors) allow the user to run commands via shell escapes, thus avoiding `sudo`'s checks. However, on most systems it is possible to prevent shell escapes with the `sudoers(5)` plugin's `noexec` functionality.

It is not meaningful to run the `cd` command directly via `sudo`, e.g.,

```
$ sudo cd /usr/local/protected
```

since when the command exits the parent process (your shell) will still be the same. Please see the **EXAMPLES** section for more information.

Running shell scripts via `sudo` can expose the same kernel bugs that make set-user-ID shell scripts unsafe on some operating systems (if your OS has a `/dev/fd/` directory, set-user-ID shell scripts are generally safe).

BUGS

If you feel you have found a bug in `sudo`, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the `sudo-users` mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

`sudo` is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with `sudo` or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo.conf — configuration for sudo front-end

DESCRIPTION

The **sudo.conf** file is used to configure the **sudo** front-end. It is used to configure sudo plugins, plugin-agnostic path names, debug flags, and other settings.

The **sudo.conf** file supports the following directives, described in detail below.

Plugin an approval, audit, I/O logging, or security policy plugin

Path a plugin-agnostic path

Set a front-end setting, such as *disable_coredump* or *group_source*

Debug debug flags to aid in debugging **sudo**, **sudoreplay**, **visudo**, and the **sudoers** plugin.

The pound sign ('#') is used to indicate a comment. Both the comment character and any text after it, up to the end of the line, are ignored.

Long lines can be continued with a backslash ('\') as the last character on the line. Note that leading white space is removed from the beginning of lines even when the continuation character is used.

Non-comment lines that don't begin with Plugin, Path, Debug, or Set are silently ignored.

The **sudo.conf** file is always parsed in the "C" locale.

Plugin configuration

sudo supports a plugin architecture for security policies and input/output logging. Third parties can develop and distribute their own policy and I/O logging plugins to work seamlessly with the **sudo** front-end. Plugins are dynamically loaded based on the contents of **sudo.conf**.

A Plugin line consists of the Plugin keyword, followed by the *symbol_name* and the *path* to the dynamic shared object that contains the plugin. The *symbol_name* is the name of the *approval_plugin*, *audit_plugin*, *io_plugin*, or *policy_plugin* struct contained in the plugin. If a plugin implements multiple plugin types, there must be a Plugin line for each unique symbol name. The *path* may be fully qualified or relative. If not fully qualified, it is relative to the directory specified by the *plugin_dir* Path setting, which defaults to /usr/libexec/sudo. In other words:

```
Plugin sudoers_policy sudoers.so
```

is equivalent to:

```
Plugin sudoers_policy /usr/libexec/sudo/sudoers.so
```

If the plugin was compiled statically into the **sudo** binary instead of being installed as a dynamic shared object, the *path* should be specified without a leading directory, as it does not actually exist in the file system. For example:

```
Plugin sudoers_policy sudoers.so
```

Starting with **sudo** 1.8.5, any additional parameters after the *path* are passed as arguments to the plugin's *open* function. For example, to override the compile-time default sudoers file mode:

```
Plugin sudoers_policy sudoers.so sudoers_mode=0440
```

See the **sudoers(5)** manual for a list of supported arguments.

The same dynamic shared object may contain multiple plugins, each with a different symbol name. The file must be owned by user-ID 0 and only writable by its owner. Because of ambiguities that arise from composite policies, only a single policy plugin may be specified. This limitation does not apply to I/O plugins.

If no **sudo.conf** file is present, or if it contains no **Plugin** lines, the **sudoers** plugin will be used as the default security policy, for I/O logging (if enabled by the policy), and for auditing. This is equivalent to the following:

```
Plugin sudoers_policy sudoers.so
Plugin sudoers_io sudoers.so
Plugin sudoers_audit sudoers.so
```

Starting with **sudo** version 1.9.1, some of the logging functionality of the **sudoers** plugin has been moved from the policy plugin to an audit plugin. To maintain compatibility with **sudo.conf** files from older **sudo** versions, if **sudoers** is configured as the security policy, it will be used as an audit plugin as well. This guarantees that the logging behavior will be consistent with that of **sudo** versions 1.9.0 and below.

For more information on the **sudo** plugin architecture, see the **sudo_plugin(5)** manual.

Path settings

A Path line consists of the Path keyword, followed by the name of the path to set and its value. For example:

```
Path intercept /usr/libexec/sudo/sudo_intercept.so
Path noexec /usr/libexec/sudo/sudo_noexec.so
Path askpass /usr/X11R6/bin/ssh-askpass
```

If no path name is specified, features relying on the specified setting will be disabled. Disabling Path settings is only supported in **sudo** version 1.8.16 and higher.

The following plugin-agnostic paths may be set in the **/etc/sudo.conf** file:

askpass	The fully qualified path to a helper program used to read the user's password when no terminal is available. This may be the case when sudo is executed from a graphical (as opposed to text-based) application. The program specified by <i>askpass</i> should display the argument passed to it as the prompt and write the user's password to the standard output. The value of <i>askpass</i> may be overridden by the SUDO_ASKPASS environment variable.
devsearch	An ordered, colon-separated search path of directories to look in for device nodes. This is used when mapping the process's tty device number to a device name on systems that do not provide such a mechanism. Sudo will <i>not</i> recurse into sub-directories. If terminal devices may be located in a sub-directory of /dev , that path must be explicitly listed in <i>devsearch</i> . The default value is /dev/pts:/dev/vt:/dev/term:/dev/zcons:/dev/pty:/dev . This option is ignored on systems that support either the devname() or _ttyname_dev() functions, for example BSD, macOS and Solaris.
intercept	The fully-qualified path to a shared library containing wrappers for the exec1() , execle() , execlp() , execv() , execve() , execvp() , and execvpe() library functions that intercepts attempts to run further commands and performs a policy check before allowing them to be executed. This is used to implement the <i>intercept</i> functionality on systems that support LD_PRELOAD or its equivalent. The default value is /usr/libexec/sudo/sudo_intercept.so .
noexec	The fully-qualified path to a shared library containing wrappers for the exec1() , execle() , execlp() , exec() , execv() , execve() , execveat() , execvp() , execvpe() , fexecve() , popen() , posix_spawn() , posix_spawnnp() , system() , and wordexp() library functions that prevent the execution of further commands. This is used to implement the <i>noexec</i> functionality on systems that support LD_PRELOAD or its equivalent. The default value is /usr/libexec/sudo/sudo_noexec.so .

plugin_dir

The default directory to use when searching for plugins that are specified without a fully qualified path name. The default value is `/usr/libexec/sudo`.

sesh

The fully-qualified path to the **sesh** binary. This setting is only used when **sudo** is built with SELinux support. The default value is `/usr/libexec/sudo/sesh`.

Other settings

The **sudo.conf** file also supports the following front-end settings:

disable_coredump

Core dumps of **sudo** itself are disabled by default to prevent the disclosure of potentially sensitive information. To aid in debugging **sudo** crashes, you may wish to re-enable core dumps by setting “`disable_coredump`” to false in **sudo.conf** as follows:

```
Set disable_coredump false
```

All modern operating systems place restrictions on core dumps from set-user-ID processes like **sudo** so this option can be enabled without compromising security. To actually get a **sudo** core file you will likely need to enable core dumps for set-user-ID processes. On BSD and Linux systems this is accomplished in the `sysctl(8)` command. On Solaris, the `coreadm(1m)` command is used to configure core dump behavior.

This setting is only available in **sudo** version 1.8.4 and higher.

developer_mode

By default **sudo** refuses to load plugins which can be modified by other than the root user. The plugin should be owned by root and write access permissions should be disabled for “group” and “other”. To make development of a plugin easier, you can disable that by setting “`developer_mode`” option to true in **sudo.conf** as follows:

```
Set developer_mode true
```

Please note that this creates a security risk, so it is not recommended on critical systems such as a desktop machine for daily use, but is intended to be used in development environments (VM, container, etc). Before enabling developer mode, ensure you understand the implications.

This setting is only available in **sudo** version 1.9.0 and higher.

group_source

sudo passes the invoking user’s group list to the policy and I/O plugins. On most systems, there is an upper limit to the number of groups that a user may belong to simultaneously (typically 16 for compatibility with NFS). On systems with the `getconf(1)` utility, running:

```
getconf NGROUPS_MAX
```

will return the maximum number of groups.

However, it is still possible to be a member of a larger number of groups--they simply won’t be included in the group list returned by the kernel for the user. Starting with **sudo** version 1.8.7, if the user’s kernel group list has the maximum number of entries, **sudo** will consult the group database directly to determine the group list. This makes it possible for the security policy to perform matching by group name even when the user is a member of more than the maximum number of groups.

The `group_source` setting allows the administrator to change this default behavior. Supported values for `group_source` are:

static	Use the static group list that the kernel returns. Retrieving the group list this way is very fast but it is subject to an upper limit as described above. It is “static” in that it does not reflect changes to the group database made after the user logs in. This was the default behavior prior to sudo 1.8.7.
dynamic	Always query the group database directly. It is “dynamic” in that changes made to the group database after the user logs in will be reflected in the group list. On some systems, querying the group database for all of a user’s groups can be time consuming when querying a network-based group database. Most operating systems provide an efficient method of performing such queries. Currently, sudo supports efficient group queries on AIX, BSD, HP-UX, Linux, macOS, and Solaris. This is the default behavior on macOS in sudo 1.9.6 and higher.
adaptive	Only query the group database if the static group list returned by the kernel has the maximum number of entries. This is the default behavior on systems other than macOS in sudo 1.8.7 and higher.

For example, to cause **sudo** to only use the kernel’s static list of groups for the user:

```
Set group_source static
```

This setting is only available in **sudo** version 1.8.7 and higher.

max_groups

The maximum number of user groups to retrieve from the group database. Values less than one or larger than 1024 will be ignored. This setting is only used when querying the group database directly. It is intended to be used on systems where it is not possible to detect when the array to be populated with group entries is not sufficiently large. By default, **sudo** will allocate four times the system’s maximum number of groups (see above) and retry with double that number if the group database query fails.

This setting is only available in **sudo** version 1.8.7 and higher. It should not be required in **sudo** versions 1.8.24 and higher and may be removed in a later release.

probe_interfaces

By default, **sudo** will probe the system’s network interfaces and pass the IP address of each enabled interface to the policy plugin. This makes it possible for the plugin to match rules based on the IP address without having to query DNS. On Linux systems with a large number of virtual interfaces, this may take a non-negligible amount of time. If IP-based matching is not required, network interface probing can be disabled as follows:

```
Set probe_interfaces false
```

This setting is only available in **sudo** version 1.8.10 and higher.

Debug settings

sudo versions 1.8.4 and higher support a flexible debugging framework that can log what **sudo** is doing internally if there is a problem.

A Debug line consists of the Debug keyword, followed by the name of the program, plugin, or shared object to debug, the debug file name, and a comma-separated list of debug flags. The debug flag syntax used by **sudo**, the **sudoers** plugin along with its associated programs and shared objects is *subsystem@priority* but a third-party plugin is free to use a different format so long as it does not include a comma (‘,’).

Examples:

```
Debug sudo /var/log/sudo_debug all@warn,plugin@info
```

would log all debugging statements at the *warn* level and higher in addition to those at the *info* level for the plugin subsystem.

```
Debug sudo_intercept.so /var/log/intercept_debug all@debug
```

would log all debugging statements, regardless of level, for the `sudo_intercept.so` shared object that implements **sudo**'s intercept functionality.

As of **sudo** 1.8.12, multiple Debug entries may be specified per program. Older versions of **sudo** only support a single Debug entry per program. Plugin-specific Debug entries are also supported starting with **sudo** 1.8.12 and are matched by either the base name of the plugin that was loaded (for example `sudoers.so`) or by the plugin's fully-qualified path name. Previously, the `sudoers` plugin shared the same Debug entry as the **sudo** front-end and could not be configured separately.

The following priorities are supported, in order of decreasing severity: *crit*, *err*, *warn*, *notice*, *diag*, *info*, *trace*, and *debug*. Each priority, when specified, also includes all priorities higher than it. For example, a priority of *notice* would include debug messages logged at *notice* and higher.

The priorities *trace* and *debug* also include function call tracing which logs when a function is entered and when it returns. For example, the following trace is for the `get_user_groups()` function located in `src/sudo.c`:

```
sudo[123] -> get_user_groups @ src/sudo.c:385
sudo[123] <- get_user_groups @ src/sudo.c:429 := groups=10,0,5
```

When the function is entered, indicated by a right arrow ‘->’, the program, process ID, function, source file, and line number are logged. When the function returns, indicated by a left arrow ‘<-’, the same information is logged along with the return value. In this case, the return value is a string.

The following subsystems are used by the **sudo** front-end:

<i>all</i>	matches every subsystem
<i>args</i>	command line argument processing
<i>conv</i>	user conversation
<i>edit</i>	<code>sudoedit</code>
<i>event</i>	event subsystem
<i>exec</i>	command execution
<i>main</i>	sudo main function
<i>netif</i>	network interface handling
<i>pcomm</i>	communication with the plugin
<i>plugin</i>	plugin configuration
<i>pty</i>	pseudo-terminal related code
<i>selinux</i>	SELinux-specific handling
<i>util</i>	utility functions
<i>utmp</i>	utmp handling

The `sudoers`(5) plugin includes support for additional subsystems.

FILES

/etc/sudo.conf	sudo front-end configuration
----------------	-------------------------------------

EXAMPLES

```

#
# Default /etc/sudo.conf file
#
# Sudo plugins:
#   Plugin plugin_name plugin_path plugin_options ...
#
# The plugin_path is relative to /usr/libexec/sudo unless
#   fully qualified.
# The plugin_name corresponds to a global symbol in the plugin
#   that contains the plugin interface structure.
# The plugin_options are optional.
#
# The sudoers plugin is used by default if no Plugin lines are present.
#Plugin sudoers_policy sudoers.so
#Plugin sudoers_io sudoers.so
#Plugin sudoers_audit sudoers.so

#
# Sudo askpass:
#   Path askpass /path/to/askpass
#
# An askpass helper program may be specified to provide a graphical
# password prompt for "sudo -A" support. Sudo does not ship with its
# own askpass program but can use the OpenSSH askpass.
#
# Use the OpenSSH askpass
#Path askpass /usr/X11R6/bin/ssh-askpass
#
# Use the Gnome OpenSSH askpass
#Path askpass /usr/libexec/openssh/gnome-ssh-askpass

#
# Sudo device search path:
#   Path devsearch /dev/path1:/dev/path2:/dev
#
# A colon-separated list of paths to check when searching for a user's
# terminal device.
#
#Path devsearch /dev/pts:/dev/vt:/dev/term:/dev/zcons:/dev/pty:/dev

#
# Sudo command interception:
#   Path intercept /path/to/sudo_intercept.so
#
# Path to a shared library containing replacements for the execv()
# and execve() library functions that perform a policy check to verify
# the command is allowed and simply return an error if not. This is
# used to implement the "intercept" functionality on systems that

```

```
# support LD_PRELOAD or its equivalent.  
#  
# The compiled-in value is usually sufficient and should only be changed  
# if you rename or move the sudo_intercept.so file.  
#  
#Path intercept /usr/libexec/sudo/sudo_intercept.so  
  
#  
# Sudo noexec:  
#   Path noexec /path/to/sudo_noexec.so  
#  
# Path to a shared library containing replacements for the execv()  
# family of library functions that just return an error. This is  
# used to implement the "noexec" functionality on systems that support  
# LD_PRELOAD or its equivalent.  
#  
# The compiled-in value is usually sufficient and should only be changed  
# if you rename or move the sudo_noexec.so file.  
#  
#Path noexec /usr/libexec/sudo/sudo_noexec.so  
  
#  
# Sudo plugin directory:  
#   Path plugin_dir /path/to/plugins  
#  
# The default directory to use when searching for plugins that are  
# specified without a fully qualified path name.  
#  
#Path plugin_dir /usr/libexec/sudo  
  
#  
# Sudo developer mode:  
#   Set developer_mode true|false  
#  
# Allow loading of plugins that are owned by non-root or are writable  
# by "group" or "other". Should only be used during plugin development.  
#Set developer_mode true  
  
#  
# Core dumps:  
#   Set disable_coredump true|false  
#  
# By default, sudo disables core dumps while it is executing (they  
# are re-enabled for the command that is run).  
# To aid in debugging sudo problems, you may wish to enable core  
# dumps by setting "disable_coredump" to false.  
#  
#Set disable_coredump false  
  
#  
# User groups:  
#   Set group_source static|dynamic|adaptive
```

```

#
# Sudo passes the user's group list to the policy plugin.
# If the user is a member of the maximum number of groups (usually 16),
# sudo will query the group database directly to be sure to include
# the full list of groups.
#
# On some systems, this can be expensive so the behavior is configurable.
# The "group_source" setting has three possible values:
#   static - use the user's list of groups returned by the kernel.
#   dynamic - query the group database to find the list of groups.
#   adaptive - if user is in less than the maximum number of groups.
#               use the kernel list, else query the group database.
#
#Set group_source static

#
# Sudo interface probing:
#   Set probe_interfaces true|false
#
# By default, sudo will probe the system's network interfaces and
# pass the IP address of each enabled interface to the policy plugin.
# On systems with a large number of virtual interfaces this may take
# a noticeable amount of time.
#
#Set probe_interfaces false

#
# Sudo debug files:
#   Debug program /path/to/debug_log subsystem@priority[,subsystem@priority]
#
# Sudo and related programs support logging debug information to a file.
# The program is typically sudo, sudoers.so, sudoreplay, or visudo.
#
# Subsystems vary based on the program; "all" matches all subsystems.
# Priority may be crit, err, warn, notice, diag, info, trace, or debug.
# Multiple subsystem@priority may be specified, separated by a comma.
#
#Debug sudo /var/log/sudo_debug all@debug
#Debug sudoers.so /var/log/sudoers_debug all@debug

```

SEE ALSO

`sudo_plugin(5)`, `sudoers(5)`, `sudo(8)`

HISTORY

See the HISTORY file in the `sudo` distribution (<https://www.sudo.ws/history.html>) for a brief history of sudo.

AUTHORS

Many people have worked on `sudo` over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo_logsrv.proto — Sudo log server protocol

DESCRIPTION

Starting with version 1.9.0, **sudo** supports sending event and I/O logs to a log server. The protocol used is written in Google’s Protocol Buffers domain specific language. The **EXAMPLES** section includes a complete description of the protocol in Protocol Buffers format.

Because there is no way to determine message boundaries when using Protocol Buffers, the wire size of each message is sent immediately preceding the message itself as a 32-bit unsigned integer in network byte order. This is referred to as “length-prefix framing” and is how Google suggests handling the lack of message delimiters.

The protocol is made up of two basic messages, *ClientMessage* and *ServerMessage*, described below. The server must accept messages up to two megabytes in size. The server may return an error if the client tries to send a message larger than two megabytes.

Client Messages

A *ClientMessage* is a container used to encapsulate all the possible message types a client may send to the server.

```
message ClientMessage {
    oneof type {
        AcceptMessage accept_msg = 1;
        RejectMessage reject_msg = 2;
        ExitMessage exit_msg = 3;
        RestartMessage restart_msg = 4;
        AlertMessage alert_msg = 5;
        IoBuffer ttyin_buf = 6;
        IoBuffer ttyout_buf = 7;
        IoBuffer stdin_buf = 8;
        IoBuffer stdout_buf = 9;
        IoBuffer stderr_buf = 10;
        ChangeWindowSize winsize_event = 11;
        CommandSuspend suspend_event = 12;
        ClientHello hello_msg = 13;
    }
}
```

The different *ClientMessage* sub-messages the client may sent to the server are described below.

TimeSpec

```
message TimeSpec {
    int64 tv_sec = 1;
    int32 tv_nsec = 2;
}
```

A *TimeSpec* is the equivalent of a POSIX struct `timespec`, containing seconds and nanoseconds members. The `tv_sec` member is a 64-bit integer to support dates after the year 2038.

InfoMessage

```
message InfoMessage {
    message StringList {
        repeated string strings = 1;
```

```

    }
  message NumberList {
    repeated int64 numbers = 1;
  }
  string key = 1;
  oneof value {
    int64 numval = 2;
    string strval = 3;
    StringList strlistval = 4;
    NumberList numlistval = 5;
  }
}

```

An *InfoMessage* is used to represent information about the invoking user as well as the execution environment the command runs in the form of key-value pairs. The key is always a string but the value may be a 64-bit integer, a string, an array of strings, or an array of 64-bit integers. The event log data is composed of *InfoMessage* entries. See the **EVENT LOG VARIABLES** section for more information.

ClientHello hello_msg

```

message ClientHello {
  string client_id = 1;
}

```

A *ClientHello* message consists of client information that may be sent to the server when the client first connects.

client_id

A free-form client description. This usually includes the name and version of the client implementation.

AcceptMessage accept_msg

```

message AcceptMessage {
  TimeSpec submit_time = 1;
  repeated InfoMessage info_msgs = 2;
  bool expect_iobufs = 3;
}

```

An *AcceptMessage* is sent by the client when a command is allowed by the security policy. It contains the following members:

submit_time

The wall clock time when the command was submitted to the security policy.

info_msgs

An array of *InfoMessage* describing the user who submitted the command as well as the execution environment of the command. This information is used to generate an event log entry and may also be used by server to determine where and how the I/O log is stored.

expect_iobufs

Set to true if the server should expect *IoBuffer* messages to follow (for I/O logging) or false if the server should only store the event log.

If an *AcceptMessage* is sent, the client must not send a *RejectMessage* or *RestartMessage*.

RejectMessage reject_msg

```
message RejectMessage {
    TimeSpec submit_time = 1;
    string reason = 2;
    repeated InfoMessage info_msgs = 3;
}
```

A *RejectMessage* is sent by the client when a command is denied by the security policy. It contains the following members:

submit_time

The wall clock time when the command was submitted to the security policy.

reason

The reason the security policy gave for denying the command.

info_msgs

An array of *InfoMessage* describing the user who submitted the command as well as the execution environment of the command. This information is used to generate an event log entry.

If a *RejectMessage* is sent, the client must not send an *AcceptMessage* or *RestartMessage*.

ExitMessage exit_msg

```
message ExitMessage {
    TimeSpec run_time = 1;
    int32 exit_value = 2;
    bool dumped_core = 3;
    string signal = 4;
    string error = 5;
}
```

An *ExitMessage* is sent by the client after the command has exited or has been terminated by a signal. It contains the following members:

run_time

The total amount of elapsed time since the command started, calculated using a monotonic clock where possible. This is not the wall clock time.

exit_value

The command's exit value in the range 0-255.

dumped_core

True if the command was terminated by a signal and dumped core.

signal

If the command was terminated by a signal, this is set to the name of the signal without the leading

“SIG”. For example, INT, TERM, KILL, SEGV.

error

A message from the client indicating that the command was terminated unexpectedly due to an error.

When performing I/O logging, the client should wait for a *commit_point* corresponding to the final *IoBuffer* before closing the connection unless the final *commit_point* has already been received.

RestartMessage restart_msg

```
message RestartMessage {
    string log_id = 1;
    TimeSpec resume_point = 2;
}
```

A *RestartMessage* is sent by the client to resume sending an existing I/O log that was previously interrupted. It contains the following members:

log_id The the server-side name for an I/O log that was previously sent to the client by the server. This may be a path name on the server or some other kind of server-side identifier.

resume_point

The point in time after which to resume the I/O log. This is in the form of a *TimeSpec* representing the amount of time since the command started, not the wall clock time. The *resume_point* should correspond to a *commit_point* previously sent to the client by the server. If the server receives a *RestartMessage* containing a *resume_point* it has not previously seen, an error will be returned to the client and the connection will be dropped.

If a *RestartMessage* is sent, the client must not send an *AcceptMessage* or *RejectMessage*.

AlertMessage alert_msg

```
message AlertMessage {
    TimeSpec alert_time = 1;
    string reason = 2;
    repeated InfoMessage info_msgs = 3;
}
```

An *AlertMessage* is sent by the client to indicate a problem detected by the security policy while the command is running that should be stored in the event log. It contains the following members:

alert_time

The wall clock time when the alert occurred.

reason The reason for the alert.

info_msgs

An optional array of *InfoMessage* describing the user who submitted the command as well as the execution environment of the command. This information is used to generate an event log entry.

IoBuffer ttyin_buf | ttyout_buf | stdin_buf | stdout_buf | stderr_buf

```
message IoBuffer {
    TimeSpec delay = 1;
    bytes data = 2;
}
```

An *IoBuffer* is used to represent data from terminal input, terminal output, standard input, standard output, or standard error. It contains the following members:

delay The elapsed time since the last record in the form of a *TimeSpec*. The *delay* should be calculated using a monotonic clock where possible.

data The binary I/O log data from terminal input, terminal output, standard input, standard output, or standard error.

ChangeWindowSize winsize_event

```
message ChangeWindowSize {
    TimeSpec delay = 1;
    int32 rows = 2;
    int32 cols = 3;
}
```

A *ChangeWindowSize* message is sent by the client when the terminal running the command changes size. It contains the following members:

- delay The elapsed time since the last record in the form of a *TimeSpec*. The *delay* should be calculated using a monotonic clock where possible.
- rows The new number of terminal rows.
- cols The new number of terminal columns.

CommandSuspend suspend_event

```
message CommandSuspend {
    TimeSpec delay = 1;
    string signal = 2;
}
```

A *CommandSuspend* message is sent by the client when the command is either suspended or resumed. It contains the following members:

- delay The elapsed time since the last record in the form of a *TimeSpec*. The *delay* should be calculated using a monotonic clock where possible.
- signal The signal name without the leading “SIG”. For example, STOP, TSTP, CONT.

Server Messages

A *ServerMessage* is a container used to encapsulate all the possible message types the server may send to a client.

```
message ServerMessage {
    oneof type {
        ServerHello hello = 1;
        TimeSpec commit_point = 2;
        string log_id = 3;
        string error = 4;
        string abort = 5;
    }
}
```

The different *ServerMessage* sub-messages the server may sent to the client are described below.

ServerHello hello

```
message ServerHello {
    string server_id = 1;
    string redirect = 2;
    repeated string servers = 3;
    bool subcommands = 4;
}
```

The *ServerHello* message consists of server information sent when the client first connects. It contains the following members:

server_id

A free-form server description. Usually this includes the name and version of the implementation running on the log server. This member is always present.

redirect

A host and port separated by a colon (:) that the client should connect to instead. The host may be a host name, an IPv4 address, or an IPv6 address in square brackets. This may be used for server load balancing. The server will disconnect after sending the *ServerHello* when it includes a **redirect**.

servers A list of other known log servers. This can be used to implement log server redundancy and allows the client to discover all other log servers simply by connecting to one known server. This member may be omitted when there is only a single log server.

subcommands

If set, the server supports logging additional commands during a session. The client may send an *AcceptMessage* or *RejectMessage* when **sudo** is running in *intercept* mode. In this mode, commands spawned from the initial command authorized by **sudo** are subject to policy restrictions and/or are logged. If *subcommands* is false, the client must not attempt to log additional commands.

TimeSpec commit_point

A periodic time stamp sent by the server to indicate when I/O log buffers have been committed to storage. This message is not sent after every *IoBuffer* but rather at a server-configurable interval. When the server receives an *ExitMessage*, it will respond with a *commit_point* corresponding to the last received *IoBuffer* before closing the connection.

string log_id

The server-side ID of the I/O log being stored, sent in response to an *AcceptMessage* where *expect_iobufs* is true.

string error

A fatal server-side error. The server will close the connection after sending the *error* message.

string abort

An *abort* message from the server indicates that the client should kill the command and terminate the session. It may be used to implement simple server-side policy. The server will close the connection after sending the *abort* message.

Protocol flow of control

The expected protocol flow is as follows:

1. Client connects to the first available server. If the client is configured to use TLS, a TLS handshake will be attempted.
2. Client sends *ClientHello*. This is currently optional but allows the server to detect a non-TLS connection on the TLS port.
3. Server sends *ServerHello*.
4. Client responds with either *AcceptMessage*, *RejectMessage*, or *RestartMessage*.
5. If client sent a *AcceptMessage* with *expect_iobufs* set, server creates a new I/O log and responds with a *log_id*.
6. Client sends zero or more *IoBuffer* messages.
7. Server periodically responds to *IoBuffer* messages with a *commit_point*.
8. Client sends an *ExitMessage* when the command exits or is killed.

9. Server sends the final *commit_point* if one is pending.
10. Server closes the connection. After receiving the final *commit_point*, the client shuts down its side of the TLS connection if TLS is in use, and closes the connection.
11. Server shuts down its side of the TLS connection if TLS is in use, and closes the connection.

At any point, the server may send an *error* or *abort* message to the client at which point the server will close the connection. If an *abort* message is received, the client should terminate the running command.

EVENT LOG VARIABLES

AcceptMessage, *AlertMessage* and *RejectMessage* classes contain an array of *InfoMessage* that should contain information about the user who submitted the command as well as information about the execution environment of the command if it was accepted.

Some variables have a *client*, *run*, or *submit* prefix. These prefixes are used to eliminate ambiguity for variables that could apply to the client program, the user submitting the command, or the command being run. Variables with a *client* prefix pertain to the program performing the connection to the log server, for example **sudo**. Variables with a *run* prefix pertain to the command that the user requested be run. Variables with a *submit* prefix pertain to the user submitting the request (the user running **sudo**).

The following *InfoMessage* entries are required:

Key	Type	Description
command	string	command that was submitted
runuser	string	name of user the command was run as
submithost	string	name of host the command was submitted on
submituser	string	name of user submitting the command

The following *InfoMessage* entries are recognized, but not required:

Key	Type	Description	
clientargv	StringList	client's original argument vector	
clientpid	int64	client's process ID	
clientppid	int64	client's parent process ID	
clientsid	int64	client's terminal session ID	
columns	int64	number of columns in the terminal	
lines	int64	number of lines in the terminal	
runargv	StringList	argument vector of command to run	
runchroot	string	root directory of command to run	
runcwd	string	running command's working directory	
runenv	StringList	the running command's environment	
rungid	int64	primary group-ID of the command	
rungids	NumberList		supplementary group-IDs for the command
rungroup	string	primary group name of the command	
rungroups	StringList	supplementary group names for the command	
runuid	int64	run user's user-ID	
submitcwd	string	submit user's current working directory	
submitenv	StringList	the submit user's environment	
submitgid	int64	submit user's primary group-ID	
submitgids	NumberList		submit user's supplementary group-IDs

```

submitgroup  string      submitting user's primary group name
submitgroups StringList submit user's supplementary group names
submituid    int64       submit user's user-ID
ttyname     string      the terminal the command was submitted from

```

The server must accept other variables not listed above but may ignore them.

EXAMPLES

The Protocol Buffers description of the log server protocol is included in full below. Note that this uses the newer “proto3” syntax.

```

syntax = "proto3";

/*
 * Client message to the server.  Messages on the wire are
 * prefixed with a 32-bit size in network byte order.
 */
message ClientMessage {
    oneof type {
        AcceptMessage accept_msg = 1;
        RejectMessage reject_msg = 2;
        ExitMessage exit_msg = 3;
        RestartMessage restart_msg = 4;
        AlertMessage alert_msg = 5;
        IoBuffer ttyin_buf = 6;
        IoBuffer ttyout_buf = 7;
        IoBuffer stdin_buf = 8;
        IoBuffer stdout_buf = 9;
        IoBuffer stderr_buf = 10;
        ChangeWindowSize winsize_event = 11;
        CommandSuspend suspend_event = 12;
    }
}

/* Equivalent of POSIX struct timespec */
message TimeSpec {
    int64 tv_sec = 1;           /* seconds */
    int32 tv_nsec = 2;          /* nanoseconds */
}

/* I/O buffer with keystroke data */
message IoBuffer {
    TimeSpec delay = 1;         /* elapsed time since last record */
    bytes data = 2;             /* keystroke data */
}

/*
 * Key/value pairs, like Privilege Manager struct info.
 * The value may be a number, a string, or a list of strings.
 */
message InfoMessage {
    message StringList {
        repeated string strings = 1;
    }
}

```

```

    }
  message NumberList {
    repeated int64 numbers = 1;
  }
  string key = 1;
  oneof value {
    int64 numval = 2;
    string strval = 3;
    StringList strlistval = 4;
    NumberList numlistval = 5;
  }
}

/*
 * Event log data for command accepted by the policy.
 */
message AcceptMessage {
  TimeSpec submit_time = 1;           /* when command was submitted */
  repeated InfoMessage info_msgs = 2; /* key,value event log data */
  bool expect_iobufs = 3;            /* true if I/O logging enabled */
}

/*
 * Event log data for command rejected by the policy.
 */
message RejectMessage {
  TimeSpec submit_time = 1;           /* when command was submitted */
  string reason = 2;                 /* reason command was rejected */
  repeated InfoMessage info_msgs = 3; /* key,value event log data */
}

/* Message sent by client when command exits. */
/* Might revisit runtime and use end_time instead */
message ExitMessage {
  TimeSpec run_time = 1;             /* total elapsed run time */
  int32 exit_value = 2;              /* 0-255 */
  bool dumped_core = 3;             /* true if command dumped core */
  string signal = 4;                /* signal name if killed by signal */
  string error = 5;                 /* if killed due to other error */
}

/* Alert message, policy module-specific. */
message AlertMessage {
  TimeSpec alert_time = 1;           /* time alert message occurred */
  string reason = 2;                 /* policy alert error string */
  repeated InfoMessage info_msgs = 3; /* key,value event log data */
}

/* Used to restart an existing I/O log on the server. */
message RestartMessage {
  string log_id = 1;                 /* ID of log being restarted */
  TimeSpec resume_point = 2;         /* resume point (elapsed time) */
}

```

```

}

/* Window size change event. */
message ChangeWindowSize {
    TimeSpec delay = 1;           /* elapsed time since last record */
    int32 rows = 2;              /* new number of rows */
    int32 cols = 3;              /* new number of columns */
}

/* Command suspend/resume event. */
message CommandSuspend {
    TimeSpec delay = 1;           /* elapsed time since last record */
    string signal = 2;            /* signal that caused suspend/resume */
}

/*
 * Server messages to the client.  Messages on the wire are
 * prefixed with a 32-bit size in network byte order.
 */
message ServerMessage {
    oneof type {
        ServerHello hello = 1;      /* server hello message */
        TimeSpec commit_point = 2;   /* cumulative time of records stored */
        string log_id = 3;          /* ID of server-side I/O log */
        string error = 4;           /* error message from server */
        string abort = 5;           /* abort message, kill command */
    }
}

/* Hello message from server when client connects. */
message ServerHello {
    string server_id = 1;          /* free-form server description */
    string redirect = 2;            /* optional redirect if busy */
    repeated string servers = 3;   /* optional list of known servers */
}

```

SEE ALSO

`sudo_logsrvd.conf(5)`, `sudoers(5)`, `sudo(8)`, `sudo_logsrvd(8)`

Protocol Buffers, <https://developers.google.com/protocol-buffers/>.

HISTORY

See the HISTORY file in the `sudo` distribution (<https://www.sudo.ws/history.html>) for a brief history of `sudo`.

AUTHORS

Many people have worked on `sudo` over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the `sudo` distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to `sudo`.

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo_logsrvd — sudo event and I/O log server

SYNOPSIS

```
sudo_logsrvd [-hnv] [-f file] [-R percentage]
```

DESCRIPTION

sudo_logsrvd is a high-performance log server that accepts event and I/O logs from **sudo**. It can be used to implement centralized logging of **sudo** logs. The server has two modes of operation: local and relay. By default, **sudo_logsrvd** stores the logs locally but it can also be configured to relay them to another server that supports the `sudo_logsrv.proto(5)` protocol.

When not relaying, event log entries may be logged either via `syslog(3)` or to a local file. I/O Logs stored locally by **sudo_logsrvd** can be replayed via the `sudoreplay(8)` utility in the same way as logs generated directly by the `sudoers` plugin.

The server also supports restarting interrupted log transfers. To distinguish completed I/O logs from incomplete ones, the I/O log timing file is set to be read-only when the log is complete.

Configuration parameters for **sudo_logsrvd** may be specified in the `sudo_logsrvd.conf(5)` file or the file specified via the **-f** option.

sudo_logsrvd rereads its configuration file when it receives SIGHUP and writes server state to the debug file (if one is configured) when it receives SIGUSR1.

The options are as follows:

-f file, --file=file

Read configuration from *file* instead of the default, `/etc/sudo_logsrvd.conf`.

-h, --help

Display a short help message to the standard output and exit.

-n, --no-fork

Run **sudo_logsrvd** in the foreground instead of detaching from the terminal and becoming a daemon.

-R percentage, --random-drop=percentage

For each message, there is a *percentage* chance that the server will drop the connection. This is only intended for debugging the ability of a client to restart a connection.

-v, --version

Print the **sudo_logsrvd** version and exit.

Securing server connections

The I/O log data sent to **sudo_logsrvd** may contain sensitive information such as passwords and should be secured using Transport Layer Security (TLS). Doing so requires having a signed certificate on the server and, if `tls_checkpeer` is enabled in `sudo_logsrvd.conf(5)`, a signed certificate on the client as well.

The certificates can either be signed by a well-known Certificate Authority (CA), or a private CA can be used. Instructions for creating a private CA are included below in the **EXAMPLES** section.

Debugging sudo_logsrvd

sudo_logsrvd supports a flexible debugging framework that is configured via `Debug` lines in the `sudo.conf(5)` file.

For more information on configuring `sudo.conf(5)`, please refer to its manual.

FILES

<code>/etc/sudo.conf</code>	Sudo front-end configuration
<code>/etc/sudo_logsrvd.conf</code>	Sudo log server configuration file
<code>/var/log/sudo_logsrvd/incoming</code>	Directory where new journals are stored when the <code>store_first_relay</code> setting is enabled.
<code>/var/log/sudo_logsrvd/outgoing</code>	Directory where completed journals are stored when the <code>store_first_relay</code> setting is enabled.
<code>/var/log/sudo-io</code>	Default I/O log file location
<code>/run/sudo/sudo_logsrvd.pid</code>	Process ID file for <code>sudo_logsrvd</code>

EXAMPLES

Creating self-signed certificates

Unless you are using certificates signed by a well-known Certificate Authority (or a local enterprise CA), you will need to create your own CA that can sign the certificates used by `sudo_logsrvd`, `sudo_sendlog`, and the `sudoers` plugin. The following steps use the `openssl(1)` command to create keys and certificates.

Initial setup

First, we need to create a directory structure to store the files for the CA. We'll create a new directory hierarchy in `/etc/ssl/sudo` for this purpose.

```
# mkdir /etc/ssl/sudo
# cd /etc/ssl/sudo
# mkdir certs csr newcerts private
# chmod 700 private
# touch index.txt
# echo 1000 > serial
```

The `serial` and `index.txt` files are used to keep track of signed certificates.

Next, we need to make a copy of the `openssl.conf` file and customize it for our new CA. The path to `openssl.cnf` is system-dependent but `/etc/ssl/openssl.cnf` is the most common location. You will need to adjust the example below if it has a different location on your system.

```
# cp /etc/ssl/openssl.cnf .
```

Now edit the `openssl.cnf` file in the current directory and make sure it contains “ca” and “CA_default” sections. Those sections should include the following settings:

```
[ ca ]
default_ca      = CA_default

[ CA_default ]
dir            = /etc/ssl/sudo
certs          = $dir/certs
database       = $dir/index.txt
```

```
certificate      = $dir/cacert.pem
serial          = $dir/serial
```

If your `openssl.conf` file already has a “CA_default” section, you may only need to modify the “dir” setting.

Creating the CA key and certificate

In order to create and sign our own certificates, we need to create a private key and a certificate for the root of the CA. First, create the private key and protect it with a pass phrase:

```
# openssl genrsa -aes256 -out private/cakey.pem 4096
# chmod 400 private/cakey.pem
```

Next, generate the root certificate, using appropriate values for the site-specific fields:

```
# openssl req -config openssl.cnf -key private/cakey.pem \
  -new -x509 -days 7300 -sha256 -extensions v3_ca \
  -out cacert.pem
```

Enter pass phrase for `private/cakey.pem`:

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank.

For some fields there will be a default value,

If you enter ‘.’, the field will be left blank.

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:Colorado

Locality Name (eg, city) []:

Organization Name (eg, company) [Internet Widgits Pty Ltd]:sudo

Organizational Unit Name (eg, section) []:sudo Certificate Authority

Common Name (e.g., server FQDN or YOUR name) []:sudo Root CA

Email Address []:

```
# chmod 444 cacert.pem
```

Finally, verify the root certificate:

```
# openssl x509 -noout -text -in cacert.pem
```

Creating and signing certificates

The server and client certificates will be signed by the previously created root CA. Usually, the root CA is not used to sign server/client certificates directly. Instead, intermediate certificates are created and signed with the root CA and the intermediate certs are used to sign CSRs (Certificate Signing Request). In this example we'll skip this part for simplicity's sake and sign the CSRs with the root CA.

First, generate the private key without a pass phrase.

```
# openssl genrsa -out private/logsrvd_key.pem 2048
# chmod 400 private/logsrvd_key.pem
```

Next, create a certificate signing request (CSR) for the server's certificate. The organization name must match the name given in the root certificate. The common name should be either the server's IP address or a fully qualified domain name.

```
# openssl req -config openssl.cnf -key private/logsrvd_key.pem -new \
             -sha256 -out csr/logsrvd_csr.pem

Enter pass phrase for private/logsrvd_key.pem:
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Colorado
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:sudo
Organizational Unit Name (eg, section) []:sudo log server
Common Name (e.g., server FQDN or YOUR name) []:logserver.example.com
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Now sign the CSR that was just created:

```
# openssl ca -config openssl.cnf -days 375 -notext -md sha256 \
             -in csr/logsrvd_csr.pem -out certs/logsrvd_cert.pem

Using configuration from openssl.cnf
Enter pass phrase for ./private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 4096 (0x1000)
    Validity
        Not Before: Nov 11 14:05:05 2019 GMT
        Not After : Nov 20 14:05:05 2020 GMT
    Subject:
        countryName          = US
        stateOrProvinceName   = Colorado
        organizationName      = sudo
        organizationalUnitName = sudo log server
        commonName            = logserve.example.com
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        Netscape Comment:
            OpenSSL Generated Certificate
        X509v3 Subject Key Identifier:
            4C:50:F9:D0:BE:1A:4C:B2:AC:90:76:56:C7:9E:16:AE:E6:9E:E5:B5
        X509v3 Authority Key Identifier:
```

```
keyid:D7:91:24:16:B1:03:06:65:1A:7A:6E:CF:51:E9:5C:CB:7A:95:3
```

```
Certificate is to be certified until Nov 20 14:05:05 2020 GMT (375 days)
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Finally, verify the new certificate:

```
# openssl verify -CAfile cacert.pem certs/ologsrvd_cert.pem
certs/ologsrvd_cert.pem: OK
```

The /etc/ssl/sudo/certs directory now contains a signed and verified certificate for use with **sudo_ologsrvd**.

To generate a client certificate, repeat the process above using a different file name.

Configuring sudo_ologsrvd to use TLS

To use TLS for client/server communication, both **sudo_ologsrvd** and the **sudoers** plugin need to be configured to use TLS. Configuring **sudo_ologsrvd** for TLS requires the following settings, assuming the same path names used earlier:

```
# Listen on port 30344 for TLS connections to any address.
listen_address = *:30344(tls)

# Path to the certificate authority bundle file in PEM format.
tls_cacert = /etc/ssl/sudo/cacert.pem

# Path to the server's certificate file in PEM format.
tls_cert = /etc/ssl/sudo/certs/ologsrvd_cert.pem

# Path to the server's private key file in PEM format.
tls_key = /etc/ssl/sudo/private/ologsrvd_key.pem
```

The root CA cert (`cacert.pem`) must be installed on the system running **sudo_ologsrvd**. If peer authentication is enabled on the client, a copy of `cacert.pem` must be present on the client system too.

SEE ALSO

`sudo.conf(5)`, `sudo_ologsrvd.conf(5)`, `sudoers(5)`, `sudo(8)`, `sudo_sendlog(8)`,
`sudoreplay(8)`

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in **sudo_ologsrvd**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo_logsrvd is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo_logsrvd.conf — configuration for sudo_logsrvd

DESCRIPTION

The **sudo_logsrvd.conf** file is used to configure the **sudo_logsrvd** log server. It uses an INI-style format made up of sections in square brackets and “key = value” pairs specific to each section below the section name. Depending on the key, values may be integers, booleans, or strings. Section and key names are not case sensitive, but values are.

The pound sign (‘#’) is used to indicate a comment. Both the comment character and any text after it, up to the end of the line, are ignored. Lines beginning with a semi-colon (‘;’) are also ignored.

Long lines can be continued with a backslash (‘\’) as the last character on the line. Note that leading white space is removed from the beginning of lines even when the continuation character is used.

The **EXAMPLES** section contains a copy of the default **sudo_logsrvd.conf** file.

The following configuration sections are recognized:

- server
- relay
- iolog
- eventlog
- syslog
- logfile

Each section is described in detail below.

server

The *server* section configures the address and port the server will listen on. The following keys are recognized:

listen_address = host[:port][(tls)]

The host name or IP address, optional port to listen on and an optional Transport Layer Security (TLS) flag in parentheses.

The host may be a host name, an IPv4 address, an IPv6 address in square brackets or the wild card entry ‘*’. A host setting of ‘*’ will cause **sudo_logsrvd** to listen on all configured network interfaces.

If the optional *tls* flag is present, **sudo_logsrvd** will secure the connection with TLS version 1.2 or 1.3. Versions of TLS prior to 1.2 are not supported. See **sudo_logsrvd(8)** for details on generating TLS keys and certificates.

If a port is specified, it may either be a port number or a known service name as defined by the system service name database. If no port is specified, port 30343 will be used for plaintext connections and port 30344 will be used for TLS connections.

The default value is:

```
listen_address = *:30343
listen_address = *:30344(tls)
```

which will listen on all configured network interfaces for both plaintext and TLS connections.

Multiple *listen_address* lines may be specified to listen on more than one port or interface.

server_log = string

Where to log server warning and error messages. Supported values are *none*, *stderr*, *syslog*, or a path name beginning with the ‘/’ character. Note that a value of *stderr* is only effective when used in conjunction with the **-n** option. The default value is *syslog*.

`pid_file = path`

The path to the file containing the process ID of the running **sudo_logsrvd**. If set to an empty value, or if **sudo_logsrvd** is run with the **-n** option, no `pid_file` will be created. If `pid_file` refers to a symbolic link, it will be ignored. The default value is `/run/sudo/sudo_logsrvd.pid`.

`tcp_keepalive = boolean`

If true, **sudo_logsrvd** will enable the TCP keepalive socket option on the client connection. This enables the periodic transmission of keepalive messages to the client. If the client does not respond to a message in time, the connection will be closed. Defaults to true.

`timeout = number`

The amount of time, in seconds, **sudo_logsrvd** will wait for the client to respond. A value of 0 will disable the timeout. The default value is 30.

`tls_cacert = path`

The path to a certificate authority bundle file, in PEM format, to use instead of the system's default certificate authority database when authenticating clients. The default is to use `/etc/ssl/sudo/cacert.pem` if it exists, otherwise the system's default certificate authority database is used.

`tls_cert = path`

The path to the server's certificate file, in PEM format. The default value is `/etc/ssl/sudo/certs/logsrvd_cert.pem`.

`tls_checkpeer = bool`

If true, client certificates will be validated by **sudo_logsrvd**; clients without a valid certificate will be unable to connect. If false, no validation of client certificates will be performed. If true and client certificates are created using a private certificate authority, the `tls_cacert` setting must be set to a CA bundle that contains the CA certificate used to generate the client certificate. The default value is `false`.

`tls_ciphers_v12 = string`

A list of ciphers to use for connections secured by TLS version 1.2 only, separated by a colon ':'. See the **CIPHER LIST FORMAT** section in `openssl-ciphers(1)` for full details. The default value is `HIGH:!aNULL` which consists of encryption cipher suites with key lengths larger than 128 bits, and some cipher suites with 128-bit keys. Cipher suites that offer no authentication are excluded.

`tls_ciphers_v13 = string`

A list of ciphers to use for connections secured by TLS version 1.3 only, separated by a colon ':'. Supported cipher suites depend on the version of OpenSSL used, but should include the following:

```
TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_CCM_SHA256
TLS_AES_128_CCM_8_SHA256
```

The default cipher suite is `TLS_AES_256_GCM_SHA384`.

`tls_dhparams = path`

The path to a file containing custom Diffie-Hellman parameters in PEM format. This file can be created with the following command:

```
openssl dhparam -out /etc/sudo_logsrvd_dhparams.pem 2048
```

By default, **sudo_logsrvd** will use the OpenSSL defaults for Diffie-Hellman key generation.

tls_key = path

The path to the server's private key file, in PEM format. The default value is `/etc/ssl/sudo/private/logsrvd_key.pem`.

tls_verify = bool

If true, **sudo_logsrvd.conf** will validate its own certificate at startup time or when the configuration is changed. If false, no verification is performed of the server certificate. When using self-signed certificates without a certificate authority, this setting should be set to false. The default value is true.

relay

The *relay* section configures the optional logsrv relay host and port the server will connect to. The TLS configuration keys are optional, by default the corresponding keys in the **server** section will be used. They are only present in this section to make it possible for the relay connection to use a different set of TLS parameters from the client-facing server. The following keys are recognized:

connect_timeout = number

The amount of time, in seconds, **sudo_logsrvd** will wait for the connection to a *relay_host* (see below) to complete. Once the connection is complete, the *timeout* setting controls the amount of time **sudo_logsrvd** will wait for the relay to respond. A value of 0 will disable the timeout. The default value is 30.

relay_dir = path

The directory in which log messages are temporarily stored before they are sent to the relay host. Messages are stored in the wire format specified by **sudo_logsrv.proto(5)**. The default value is `/var/log/sudo_logsrvd`.

relay_host = host[:port][(tls)]

The relay host name or IP address, optional port to connect to and an optional Transport Layer Security (TLS) flag in parentheses. The syntax is identical to *listen_address* in the **server** section with one exception: the wild card '*' syntax is not supported.

When this setting is enabled, messages from the client will be forwarded to one of the specified relay hosts instead of being stored locally. The *host* could be running an instance of **sudo_logsrvd** or another server that supports the **sudo_logsrv.proto(5)** protocol.

If multiple *relay_host* lines are specified, the first available relay host will be used.

retry_interval = number

The number of seconds to wait after a connection error before making a new attempt to forward a message to a relay host. The default value is 30 seconds.

store_first = boolean

If true, **sudo_logsrvd** will store logs locally before relaying them. Once the log is complete, a connection to the relay host is opened and the log is relayed. If the network connection is interrupted before the log can be fully transferred, it will be retransmitted later. The default is to relay logs in real-time.

tcp_keepalive = boolean

If true, **sudo_logsrvd** will enable the TCP keepalive socket option on the relay connection. This enables the periodic transmission of keepalive messages to the relay server. If the relay does not respond to a message in time, the connection will be closed.

timeout = number

The amount of time, in seconds, **sudo_logsrvd** will wait for the relay server to respond after a connection has succeeded. A value of 0 will disable the timeout. The default value is 30.

tls_cacert = path

The path to a certificate authority bundle file, in PEM format, to use instead of the system's default certificate authority database when authenticating clients. The default is to use the value specified in the **server** section, or the system's default certificate authority database if no value is set.

tls_cert = path

The path to the server's certificate file, in PEM format. The default is to use the value specified in the **server** section.

tls_checkpeer = bool

If true, the relay host's certificate will be validated by **sudo_logsrvd**; connections to a relay without a valid certificate will fail. If false, no validation of relay certificates will be performed. If true and relay certificates are created using a private certificate authority, the **tls_cacert** setting must be set to a CA bundle that contains the CA certificate used to generate the relay certificate. The default is to use the value specified in the **server** section.

tls_ciphers_v12 = string

A list of ciphers to use for connections secured by TLS version 1.2 only, separated by a colon ':'. See the **CIPHER LIST FORMAT** section in **openssl-ciphers(1)** for full details. The default is to use the value specified in the **server** section.

tls_ciphers_v13 = string

A list of ciphers to use for connections secured by TLS version 1.3 only, separated by a colon ':'. Supported cipher suites depend on the version of OpenSSL used, see the **server** section for more information. The default is to use the value specified in the **server** section.

tls_dhparams = path

The path to a file containing custom Diffie-Hellman parameters in PEM format. The default is to use the value specified in the **server** section.

tls_key = path

The path to the server's private key file, in PEM format. The default is to use the value specified in the **server** section.

tls_verify = bool

If true, the server's certificate used for relaying will be verified at startup. If false, no verification is performed of the server certificate. When using self-signed certificates without a certificate authority, this setting should be set to false. The default is to use the value specified in the **server** section.

ilog

The **ilog** section configures I/O log parameters. These settings are identical to the I/O configuration in **sudoers(5)**. The following keys are recognized:

ilog_compress = boolean

If set, I/O logs will be compressed using **zlib**. Enabling compression can make it harder to view the logs in real-time as the program is executing due to buffering. The default value is **false**.

ilog_dir = path

The top-level directory to use when constructing the path name for the I/O log directory. The session sequence number, if any, is stored in the directory. The default value is **/var/log/sudo-io**.

The following percent ('%') escape sequences are supported:

- %{seq}
expanded to a monotonically increasing base-36 sequence number, such as 0100A5, where every two digits are used to form a new directory, e.g., 01/00/A5
- %{user}
expanded to the invoking user's login name
- %{group}
expanded to the name of the invoking user's real group-ID
- %{runas_user}
expanded to the login name of the user the command will be run as (e.g., root)
- %{runas_group}
expanded to the group name of the user the command will be run as (e.g., wheel)
- %{hostname}
expanded to the local host name without the domain name
- %{command}
expanded to the base name of the command being run

In addition, any escape sequences supported by the system's `strftime(3)` function will be expanded.

To include a literal '%' character, the string '%%' should be used.

iolog_file = path

The path name, relative to *iolog_dir*, in which to store I/O logs. Note that *iolog_file* may contain directory components. The default value is %{seq}.

See the *iolog_dir* setting above for a list of supported percent ('%') escape sequences.

In addition to the escape sequences, path names that end in six or more Xs will have the Xs replaced with a unique combination of digits and letters, similar to the `mkttemp(3)` function.

If the path created by concatenating *iolog_dir* and *iolog_file* already exists, the existing I/O log file will be truncated and overwritten unless *iolog_file* ends in six or more Xs.

iolog_flush = boolean

If set, I/O log data is flushed to disk after each write instead of buffering it. This makes it possible to view the logs in real-time as the program is executing but may significantly reduce the effectiveness of I/O log compression. I/O logs are always flushed before sending a commit point to the client regardless of this setting. The default value is true.

iolog_group = name

The group name to look up when setting the group-ID on new I/O log files and directories. If *iolog_group* is not set, the primary group-ID of the user specified by *iolog_user* is used. If neither *iolog_group* nor *iolog_user* are set, I/O log files and directories are created with group-ID 0.

iolog_mode = mode

The file mode to use when creating I/O log files. Mode bits for read and write permissions for owner, group, or other are honored, everything else is ignored. The file permissions will always include the owner read and write bits, even if they are not present in the specified mode. When creating I/O log directories, search (execute) bits are added to match the read and write bits specified by *iolog_mode*. The default value is 0600.

iolog_user = name

The user name to look up when setting the owner of new I/O log files and directories. If *iolog_group* is set, it will be used instead of the user's primary group-ID. By default, I/O log files and directories are created with user and group-ID 0.

maxseq = number

The maximum sequence number that will be substituted for the "%{seq}" escape in the I/O log file (see the *iolog_dir* description above for more information). While the value substituted for "%{seq}" is in base 36, *maxseq* itself should be expressed in decimal. Values larger than 2176782336 (which corresponds to the base 36 sequence number "ZZZZZZ") will be silently truncated to 2176782336. The default value is 2176782336.

eventlog

The *eventlog* section configures how (and if) security policy events are logged.

log_type = string

Where to log accept, reject, and alert events reported by the policy. Supported values are *syslog*, *logfile*, and *none*. The default value is *syslog*.

log_exit = boolean

If true, **sudo_logsrvd** will log an event when a command exits or is terminated by a signal. Defaults to false.

log_format = string

The event log format. Supported log formats are "sudo" for traditional sudo-style logs and "json" for JSON-format logs. The JSON log entries contain the full contents of the accept, reject, exit and alert messages. The default value is *sudo*.

syslog

The *syslog* section configures how events are logged via *syslog(3)*.

facility = string

Syslog facility if syslog is being used for logging. Defaults to authpriv.

The following syslog facilities are supported: **authpriv** (if your OS supports it), **auth**, **daemon**, **user**, **local0**, **local1**, **local2**, **local3**, **local4**, **local5**, **local6**, and **local7**.

accept_priority = string

Syslog priority to use when the user is allowed to run a command and authentication is successful. Defaults to notice.

The following syslog priorities are supported: **alert**, **crit**, **debug**, **emerg**, **err**, **info**, **notice**, **warning**, and **none**. Setting it to a value of **none** will disable logging of successful commands.

reject_priority = string

Syslog priority to use when the user is not allowed to run a command or when authentication is unsuccessful. Defaults to alert.

See *accept_priority* for the list of supported syslog priorities.

alert_priority = string

Syslog priority to use for event log alert messages received from the client. Defaults to alert.

See *accept_priority* for the list of supported syslog priorities.

maxlen = number

On many systems, *syslog(3)* has a relatively small log buffer. IETF RFC 5424 states that syslog servers must support messages of at least 480 bytes and should support messages up to 2048 bytes. By

default, **sudo_logsrvd** creates log messages up to 960 bytes which corresponds to the historic BSD syslog implementation which used a 1024 byte buffer to store the message, date, hostname, and program name.

To prevent syslog messages from being truncated, **sudo_logsrvd** will split up sudo-style log messages that are larger than *maxlen* bytes. When a message is split, additional parts will include the string “(command continued)” after the user name and before the continued command line arguments. JSON-format log entries are never split and are not affected by *maxlen*.

server_facility = string

Syslog facility if syslog is being used for server warning messages. See above for a list of supported facilities. Defaults to daemon

logfile

The *logfile* section consists of settings related to logging to a plain file (not syslog).

path = string

The path to the file-based event log. This path must be fully-qualified and start with a ‘/’ character. The default value is /var/log/sudo.log.

time_format = string

The string used when formatting the date and time for file-based event logs. Formatting is performed via the system’s strftime(3) function so any escape sequences supported by that function will be expanded. The default value is “%h %e %T” which produces dates like “Oct 3 07:15:24” in the C locale.

FILES

/etc/sudo_logsrvd.conf
Sudo log server configuration file

EXAMPLES

```
#  
# sudo logsrv daemon configuration  
  
[server]  
# The host name or IP address and port to listen on with an optional TLS  
# flag. If no port is specified, port 30343 will be used for plaintext  
# connections and port 30344 will be used to TLS connections.  
# The following forms are accepted:  
#   listen_address = hostname(tls)  
#   listen_address = hostname:port(tls)  
#   listen_address = IPv4_address(tls)  
#   listen_address = IPv4_address:port(tls)  
#   listen_address = [IPv6_address](tls)  
#   listen_address = [IPv6_address]:port(tls)  
#  
# The (tls) suffix should be omitted for plaintext connections.  
#  
# Multiple listen_address settings may be specified.  
# The default is to listen on all addresses.  
#listen_address = *:30343  
#listen_address = *:30344(tls)
```

```
# The file containing the ID of the running sudo_logsrvd process.  
#pid_file = /run/sudo/sudo_logsrvd.pid  
  
# Where to log server warnings: none, stderr, syslog, or a path name.  
#server_log = syslog  
  
# If true, enable the SO_KEEPALIVE socket option on client connections.  
# Defaults to true.  
#tcp_keepalive = true  
  
# The amount of time, in seconds, the server will wait for the client to  
# respond. A value of 0 will disable the timeout. The default value is 30.  
#timeout = 30  
  
# If true, the server will validate its own certificate at startup.  
# Defaults to true.  
#tls_verify = true  
  
# If true, client certificates will be validated by the server;  
# clients without a valid certificate will be unable to connect.  
# By default, client certs are not checked.  
#tls_checkpeer = false  
  
# Path to a certificate authority bundle file in PEM format to use  
# instead of the system's default certificate authority database.  
#tls_cacert = /etc/ssl/sudo/cacert.pem  
  
# Path to the server's certificate file in PEM format.  
# Required for TLS connections.  
#tls_cert = /etc/ssl/sudo/certs/logsrvd_cert.pem  
  
# Path to the server's private key file in PEM format.  
# Required for TLS connections.  
#tls_key = /etc/ssl/sudo/private/logsrvd_key.pem  
  
# TLS cipher list (see "CIPHER LIST FORMAT" in the openssl-ciphers manual).  
# NOTE that this setting is only effective if the negotiated protocol  
# is TLS version 1.2.  
# The default cipher list is HIGH:!aNULL.  
#tls_ciphers_v12 = HIGH:!aNULL  
  
# TLS cipher list if the negotiated protocol is TLS version 1.3.  
# The default cipher list is TLS_AES_256_GCM_SHA384.  
#tls_ciphers_v13 = TLS_AES_256_GCM_SHA384  
  
# Path to the Diffie-Hellman parameter file in PEM format.  
# If not set, the server will use the OpenSSL defaults.  
#tls_dhparams = /etc/ssl/sudo/logsrvd_dhparams.pem  
  
[relay]  
# The host name or IP address and port to send logs to in relay mode.  
# The syntax is identical to listen_address with the exception of
```

```
# the wild card ('*') syntax. When this setting is enabled, logs will
# be relayed to the specified host instead of being stored locally.
# This setting is not enabled by default.
#relay_host = relayhost.dom.ain
#relay_host = relayhost.dom.ain(tls)

# The amount of time, in seconds, the server will wait for a connection
# to the relay server to complete. A value of 0 will disable the timeout.
# The default value is 30.
#connect_timeout = 30

# The directory to store messages in before they are sent to the relay.
# Messages are stored in wire format.
# The default value is /var/log/sudo_logsrvd.
#relay_dir = /var/log/sudo_logsrvd

# The number of seconds to wait after a connection error before
# making a new attempt to forward a message to a relay host.
# The default value is 30.
#retry_interval = 30

# Whether to store the log before relaying it. If true, enable store
# and forward mode. If false, the client connection is immediately
# relayed. Defaults to false.
#store_first = true

# If true, enable the SO_KEEPALIVE socket option on relay connections.
# Defaults to true.
#tcp_keepalive = true

# The amount of time, in seconds, the server will wait for the relay to
# respond. A value of 0 will disable the timeout. The default value is 30.
#timeout = 30

# If true, the server's relay certificate will be verified at startup.
# The default is to use the value in the [server] section.
#tls_verify = true

# Whether to verify the relay's certificate for TLS connections.
# The default is to use the value in the [server] section.
#tls_checkpeer = false

# Path to a certificate authority bundle file in PEM format to use
# instead of the system's default certificate authority database.
# The default is to use the value in the [server] section.
#tls_cacert = /etc/ssl/sudo/cacert.pem

# Path to the server's certificate file in PEM format.
# The default is to use the certificate in the [server] section.
#tls_cert = /etc/ssl/sudo/certs/logsrvd_cert.pem

# Path to the server's private key file in PEM format.
```

```

# The default is to use the key in the [server] section.
#tls_key = /etc/ssl/sudo/private/logsrvd_key.pem

# TLS cipher list (see "CIPHER LIST FORMAT" in the openssl-ciphers manual).
# NOTE that this setting is only effective if the negotiated protocol
# is TLS version 1.2.
# The default is to use the value in the [server] section.
#tls_ciphers_v12 = HIGH:!aNULL

# TLS cipher list if the negotiated protocol is TLS version 1.3.
# The default is to use the value in the [server] section.
#tls_ciphers_v13 = TLS_AES_256_GCM_SHA384

# Path to the Diffie-Hellman parameter file in PEM format.
# The default is to use the value in the [server] section.
#tls_dhparams = /etc/ssl/sudo/logsrvd_dhparams.pem

[iolog]
# The top-level directory to use when constructing the path name for the
# I/O log directory. The session sequence number, if any, is stored here.
#iolog_dir = /var/log/sudo-io

# The path name, relative to iolog_dir, in which to store I/O logs.
# Note that iolog_file may contain directory components.
#iolog_file = %{seq}

# If set, I/O logs will be compressed using zlib. Enabling compression can
# make it harder to view the logs in real-time as the program is executing.
#iolog_compress = false

# If set, I/O log data is flushed to disk after each write instead of
# buffering it. This makes it possible to view the logs in real-time
# as the program is executing but reduces the effectiveness of compression.
#iolog_flush = true

# The group to use when creating new I/O log files and directories.
# If iolog_group is not set, the primary group-ID of the user specified
# by iolog_user is used. If neither iolog_group nor iolog_user
# are set, I/O log files and directories are created with group-ID 0.
#iolog_group = wheel

# The user to use when setting the user-ID and group-ID of new I/O
# log files and directories. If iolog_group is set, it will be used
# instead of the user's primary group-ID. By default, I/O log files
# and directories are created with user and group-ID 0.
#iolog_user = root

# The file mode to use when creating I/O log files. The file permissions
# will always include the owner read and write bits, even if they are
# not present in the specified mode. When creating I/O log directories,
# search (execute) bits are added to match the read and write bits
# specified by iolog_mode.

```

```

#ilog_mode = 0600

# The maximum sequence number that will be substituted for the "%{seq}"
# escape in the I/O log file. While the value substituted for "%{seq}"
# is in base 36, maxseq itself should be expressed in decimal. Values
# larger than 2176782336 (which corresponds to the base 36 sequence
# number "ZZZZZZ") will be silently truncated to 2176782336.
#maxseq = 2176782336

[eventlog]
# Where to log accept, reject, exit, and alert events.
# Accepted values are syslog, logfile, or none.
# Defaults to syslog
#log_type = syslog

# Whether to log an event when a command exits or is terminated by a signal.
# Defaults to false
#log_exit = true

# Event log format.
# Currently only sudo-style event logs are supported.
#log_format = sudo

[syslog]
# The maximum length of a syslog payload.
# On many systems, syslog(3) has a relatively small log buffer.
# IETF RFC 5424 states that syslog servers must support messages
# of at least 480 bytes and should support messages up to 2048 bytes.
# Messages larger than this value will be split into multiple messages.
# maxlen = 960

# The syslog facility to use for event log messages.
# The following syslog facilities are supported: authpriv (if your OS
# supports it), auth, daemon, user, local0, local1, local2, local3,
# local4, local5, local6, and local7.
#facility = authpriv

# Syslog priority to use for event log accept messages, when the command
# is allowed by the security policy. The following syslog priorities are
# supported: alert, crit, debug, emerg, err, info, notice, warning, none.
#accept_priority = notice

# Syslog priority to use for event log reject messages, when the command
# is not allowed by the security policy.
#reject_priority = alert

# Syslog priority to use for event log alert messages reported by the
# client.
#alert_priority = alert

# The syslog facility to use for server warning messages.
# Defaults to daemon.

```

```
#server_facility = daemon

[logfile]
# The path to the file-based event log.
# This path must be fully-qualified and start with a '/' character.
#path = /var/log/sudo

# The format string used when formatting the date and time for
# file-based event logs. Formatting is performed via strftime(3) so
# any format string supported by that function is allowed.
#time_format = %h %e %T
```

SEE ALSO

`strftime(3)`, `sudo.conf(5)`, `sudoers(5)`, `sudo(8)`, `sudo_logsrvd(8)`

HISTORY

See the HISTORY file in the `sudo` distribution (<https://www.sudo.ws/history.html>) for a brief history of `sudo`.

AUTHORS

Many people have worked on `sudo` over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the `sudo` distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to `sudo`.

BUGS

If you feel you have found a bug in `sudo`, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

`sudo` is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with `sudo` or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo_plugin — Sudo Plugin API

DESCRIPTION

Starting with version 1.8, **sudo** supports a plugin API for policy and session logging. Plugins may be compiled as dynamic shared objects (the default on systems that support them) or compiled statically into the **sudo** binary itself. By default, the **sudoers** plugin provides audit, security policy and I/O logging capabilities. Via the plugin API, **sudo** can be configured to use alternate plugins provided by third parties. The plugins to be used are specified in the **sudo.conf(5)** file.

The API is versioned with a major and minor number. The minor version number is incremented when additions are made. The major number is incremented when incompatible changes are made. A plugin should be check the version passed to it and make sure that the major version matches.

The plugin API is defined by the **sudo_plugin.h** header file.

Policy plugin API

A policy plugin must declare and populate a **policy_plugin** struct in the global scope. This structure contains pointers to the functions that implement the **sudo** policy checks. The name of the symbol should be specified in **sudo.conf(5)** along with a path to the plugin so that **sudo** can load it.

```
struct policy_plugin {
#define SUDO_POLICY_PLUGIN 1
    unsigned int type; /* always SUDO_POLICY_PLUGIN */
    unsigned int version; /* always SUDO_API_VERSION */
    int (*open)(unsigned int version, sudo_conv_t conversation,
                sudo_printf_t plugin_printf, char * const settings[],
                char * const user_info[], char * const user_env[],
                char * const plugin_options[], const char **errstr);
    void (*close)(int exit_status, int error);
    int (*show_version)(int verbose);
    int (*check_policy)(int argc, char * const argv[],
                        char *env_add[], char **command_info[],
                        char **argv_out[], char **user_env_out[], const char **errstr);
    int (*list)(int argc, char * const argv[], int verbose,
                const char *list_user, const char **errstr);
    int (*validate)(const char **errstr);
    void (*invalidate)(int remove);
    int (*init_session)(struct passwd *pwd, char **user_env[],
                        const char **errstr);
    void (*register_hooks)(int version,
                          int (*register_hook)(struct sudo_hook *hook));
    void (*deregister_hooks)(int version,
                            int (*deregister_hook)(struct sudo_hook *hook));
    struct sudo_plugin_event * (*event_alloc)(void);
};
```

The **policy_plugin** struct has the following fields:

type The **type** field should always be set to **SUDO_POLICY_PLUGIN**.

version

The **version** field should be set to **SUDO_API_VERSION**.

This allows **sudo** to determine the API version the plugin was built against.

open

```
int (*open)(unsigned int version, sudo_conv_t conversation,
            sudo_printf_t plugin_printf, char * const settings[],
            char * const user_info[], char * const user_env[],
            char * const plugin_options[], const char **errstr);
```

Returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error. In the latter case, **sudo** will print a usage message before it exits. If an error occurs, the plugin may optionally call the **conversation()** or **plugin_printf()** function with **SUDO_CONF_ERROR_MSG** to present additional error information to the user.

The function arguments are as follows:

version

The version passed in by **sudo** allows the plugin to determine the major and minor version number of the plugin API supported by **sudo**.

conversation

A pointer to the **conversation()** function that can be used by the plugin to interact with the user (see **Conversation API** for details). Returns 0 on success and -1 on failure.

plugin_printf

A pointer to a **printf()**-style function that may be used to display informational or error messages (see **Conversation API** for details). Returns the number of characters printed on success and -1 on failure.

settings

A vector of user-supplied **sudo** settings in the form of “name=value” strings. The vector is terminated by a NULL pointer. These settings correspond to options the user specified when running **sudo**. As such, they will only be present when the corresponding option has been specified on the command line.

When parsing *settings*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

The following values may be set by **sudo**:

bsdauth_type=string

Authentication type, if specified by the **-a** option, to use on systems where BSD authentication is supported.

closefrom=number

If specified, the user has requested via the **-C** option that **sudo** close all files descriptors with a value of *number* or higher. The plugin may optionally pass this, or another value, back in the *command_info* list.

cmnd_chroot=string

The root directory (see **chroot(2)**) to run the command in, as specified by the user via the **-R** option. The plugin may ignore or restrict the user’s ability to specify a new root directory. Only available starting with API version 1.16.

cmnd_cwd=string

The working directory to run the command in, as specified by the user via the **-D** option. The plugin may ignore or restrict the user’s ability to specify a new working directory. Only available starting with API version 1.16.

debug_flags=string

A debug file path name followed by a space and a comma-separated list of debug flags that correspond to the plugin's Debug entry in sudo.conf(5), if there is one. The flags are passed to the plugin exactly as they appear in sudo.conf(5). The syntax used by **sudo** and the **sudoers** plugin is *subsystem@priority* but a plugin is free to use a different format so long as it does not include a comma (‘,’). Prior to **sudo** 1.8.12, there was no way to specify plugin-specific *debug_flags* so the value was always the same as that used by the **sudo** front-end and did not include a path name, only the flags themselves. As of version 1.7 of the plugin interface, **sudo** will only pass *debug_flags* if sudo.conf(5) contains a plugin-specific Debug entry.

ignore_ticket=bool

Set to true if the user specified the **-k** option along with a command, indicating that the user wishes to ignore any cached authentication credentials. *implied_shell* to true. This allows **sudo** with no arguments to be used similarly to su(1). If the plugin does not support this usage, it may return a value of -2 from the **check_policy()** function, which will cause **sudo** to print a usage message and exit.

implied_shell=bool

If the user does not specify a program on the command line, **sudo** will pass the plugin the path to the user's shell and set

login_class=string

BSD login class to use when setting resource limits and nice value, if specified by the **-c** option.

login_shell=bool

Set to true if the user specified the **-i** option, indicating that the user wishes to run a login shell.

max_groups=int

The maximum number of groups a user may belong to. This will only be present if there is a corresponding setting in sudo.conf(5).

network_addrs=list

A space-separated list of IP network addresses and netmasks in the form “addr/netmask”, e.g., “192.168.1.2/255.255.255.0”. The address and netmask pairs may be either IPv4 or IPv6, depending on what the operating system supports. If the address contains a colon (‘:’), it is an IPv6 address, else it is IPv4.

noninteractive=bool

Set to true if the user specified the **-n** option, indicating that **sudo** should operate in non-interactive mode. The plugin may reject a command run in non-interactive mode if user interaction is required.

plugin_dir=string

The default plugin directory used by the **sudo** front-end. This is the default directory set at compile time and may not correspond to the directory the running plugin was loaded from. It may be used by a plugin to locate support files.

plugin_path=string

The path name of plugin loaded by the **sudo** front-end. The path name will be a fully-qualified unless the plugin was statically compiled into **sudo**.

preserve_environment=bool

Set to true if the user specified the **-E** option, indicating that the user wishes to preserve the environment.

preserve_groups=bool
Set to true if the user specified the **-P** option, indicating that the user wishes to preserve the group vector instead of setting it based on the runas user.

progname=string
The command name that sudo was run as, typically “sudo” or “sudoedit”.

prompt=string
The prompt to use when requesting a password, if specified via the **-p** option.

remote_host=string
The name of the remote host to run the command on, if specified via the **-h** option. Support for running the command on a remote host is meant to be implemented via a helper program that is executed in place of the user-specified command. The **sudo** front-end is only capable of executing commands on the local host. Only available starting with API version 1.4.

run_shell=bool
Set to true if the user specified the **-s** option, indicating that the user wishes to run a shell.

runas_group=string
The group name or group-ID to run the command as, if specified via the **-g** option.

runas_user=string
The user name or user-ID to run the command as, if specified via the **-u** option.

selinux_role=string
SELinux role to use when executing the command, if specified by the **-r** option.

selinux_type=string
SELinux type to use when executing the command, if specified by the **-t** option.

set_home=bool
Set to true if the user specified the **-H** option. If true, set the HOME environment variable to the target user’s home directory.

sudoedit=bool
Set to true when the **-e** option is specified or if invoked as **sudoedit**. The plugin shall substitute an editor into *argv* in the **check_policy()** function or return -2 with a usage error if the plugin does not support *sudoedit*. For more information, see the *check_policy* section.

timeout=string
Command timeout specified by the user via the **-T** option. Not all plugins support command timeouts and the ability of the user to set a timeout may be restricted by policy. The format of the timeout string is plugin-specific.

Additional settings may be added in the future so the plugin should silently ignore settings that it does not recognize.

user_info
A vector of information about the user running the command in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *user_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

The following values may be set by **sudo**:

cols=int

The number of columns the user's terminal supports. If there is no terminal device available, a default value of 80 is used.

cwd=string

The user's current working directory.

egid=gid_t

The effective group-ID of the user invoking **sudo**.

euid=uid_t

The effective user-ID of the user invoking **sudo**.

gid=gid_t

The real group-ID of the user invoking **sudo**.

groups=list

The user's supplementary group list formatted as a string of comma-separated group-IDs.

host=string

The local machine's hostname as returned by the `gethostname(2)` system call.

lines=int

The number of lines the user's terminal supports. If there is no terminal device available, a default value of 24 is used.

pgid=int

The ID of the process group that the running **sudo** process is a member of. Only available starting with API version 1.2.

pid=int

The process ID of the running **sudo** process. Only available starting with API version 1.2.

ppid=int

The parent process ID of the running **sudo** process. Only available starting with API version 1.2.

rlimit_as=soft,hard

The maximum size to which the process's address space may grow (in bytes), if supported by the operating system. The soft and hard limits are separated by a comma. A value of "infinity" indicates that there is no limit. Only available starting with API version 1.16.

rlimit_core=soft,hard

The largest size core dump file that may be created (in bytes). The soft and hard limits are separated by a comma. A value of "infinity" indicates that there is no limit. Only available starting with API version 1.16.

rlimit_cpu=soft,hard

The maximum amount of CPU time that the process may use (in seconds). The soft and hard limits are separated by a comma. A value of "infinity" indicates that there is no limit. Only available starting with API version 1.16.

rlimit_data=soft,hard

The maximum size of the data segment for the process (in bytes). The soft and hard limits are separated by a comma. A value of "infinity" indicates that there is no limit. Only available starting with API version 1.16.

rlimit_fsize=soft,hard

The largest size file that the process may create (in bytes). The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_locks=soft,hard

The maximum number of locks that the process may establish, if supported by the operating system. The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_memlock=soft,hard

The maximum size that the process may lock in memory (in bytes), if supported by the operating system. The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_nofile=soft,hard

The maximum number of files that the process may have open. The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_nproc=soft,hard

The maximum number of processes that the user may run simultaneously. The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_rss=soft,hard

The maximum size to which the process’s resident set size may grow (in bytes). The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

rlimit_stack=soft,hard

The maximum size to which the process’s stack may grow (in bytes). The soft and hard limits are separated by a comma. A value of “infinity” indicates that there is no limit. Only available starting with API version 1.16.

sid=int

The session ID of the running **sudo** process or 0 if **sudo** is not part of a POSIX job control session. Only available starting with API version 1.2.

tcpgid=int

The ID of the foreground process group associated with the terminal device associated with the **sudo** process or 0 if there is no terminal present. Only available starting with API version 1.2.

tty=string

The path to the user’s terminal device. If the user has no terminal device associated with the session, the value will be empty, as in “`tty=`”.

uid=uid_t

The real user-ID of the user invoking **sudo**.

umask=octal

The invoking user’s file creation mask. Only available starting with API version 1.10.

user=string

The name of the user invoking **sudo**.

user_env

The user's environment in the form of a NULL-terminated vector of "name=value" strings.

When parsing *user_env*, the plugin should split on the **first** equal sign ('=') since the *name* field will never include one itself but the *value* might.

plugin_options

Any (non-comment) strings immediately after the plugin path are passed as arguments to the plugin. These arguments are split on a white space boundary and are passed to the plugin in the form of a NULL-terminated array of strings. If no arguments were specified, *plugin_options* will be the NULL pointer.

NOTE: the *plugin_options* parameter is only available starting with API version 1.2. A plugin **must** check the API version specified by the **sudo** front-end before using *plugin_options*. Failure to do so may result in a crash.

errstr

If the **open()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

close

```
void (*close)(int exit_status, int error);
```

The **close()** function is called when **sudo** is finished, shortly before it exits. Starting with API version 1.15, **close()** is called regardless of whether or not a command was actually executed. This makes it possible for plugins to perform cleanup even when a command was not run. It is not possible to tell whether a command was run based solely on the arguments passed to the **close()** function. To determine if a command was actually run, the plugin must keep track of whether or not the **check_policy()** function returned successfully.

The function arguments are as follows:

exit_status

The command's exit status, as returned by the **wait(2)** system call, or zero if no command was run. The value of *exit_status* is undefined if *error* is non-zero.

error If the command could not be executed, this is set to the value of **errno** set by the **execve(2)** system call. The plugin is responsible for displaying error information via the **conversation()** or **plugin_printf()** function. If the command was successfully executed, the value of *error* is zero.

If no **close()** function is defined, no I/O logging plugins are loaded, and neither the **timeout** nor **use_pty** options are set in the **command_info** list, the **sudo** front-end may execute the command directly instead of running it as a child process.

show_version

```
int (*show_version)(int verbose);
```

The **show_version()** function is called by **sudo** when the user specifies the **-v** option. The plugin may display its version information to the user via the **conversation()** or **plugin_printf()** function using **SUDO_CONV_INFO_MSG**. If the user requests detailed version information, the *verbose* flag will be set.

Returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error, although the return value is currently ignored.

check_policy

```
int (*check_policy)(int argc, char * const argv[], char *env_add[],
    char **command_info[], char **argv_out[], char **user_env_out[],
    const char **errstr);
```

The **check_policy()** function is called by **sudo** to determine whether the user is allowed to run the specified commands.

If the *sudoedit* option was enabled in the *settings* array passed to the **open()** function, the user has requested *sudoedit* mode. *sudoedit* is a mechanism for editing one or more files where an editor is run with the user's credentials instead of with elevated privileges. **sudo** achieves this by creating user-writable temporary copies of the files to be edited and then overwriting the originals with the temporary copies after editing is complete. If the plugin supports *sudoedit*, it should choose the editor to be used, potentially from a variable in the user's environment, such as `EDITOR`, and include it in *argv_out* (note that environment variables may include command line options). The files to be edited should be copied from *argv* into *argv_out*, separated from the editor and its arguments by a “--” element. The “--” will be removed by **sudo** before the editor is executed. The plugin should also set *sudoedit=true* in the *command_info* list.

The **check_policy()** function returns 1 if the command is allowed, 0 if not allowed, -1 for a general error, or -2 for a usage error or if *sudoedit* was specified but is unsupported by the plugin. In the latter case, **sudo** will print a usage message before it exits. If an error occurs, the plugin may optionally call the **conversation()** or **plugin_printf()** function with `SUDO_CONF_ERROR_MSG` to present additional error information to the user.

The function arguments are as follows:

argc The number of elements in *argv*, not counting the final NULL pointer.

argv The argument vector describing the command the user wishes to run, in the same form as what would be passed to the `execve(2)` system call. The vector is terminated by a NULL pointer.

env_add

Additional environment variables specified by the user on the command line in the form of a NULL-terminated vector of “name=value” strings. The plugin may reject the command if one or more variables are not allowed to be set, or it may silently ignore such variables.

When parsing *env_add*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

command_info

Information about the command being run in the form of “name=value” strings. These values are used by **sudo** to set the execution environment when running a command. The plugin is responsible for creating and populating the vector, which must be terminated with a NULL pointer. The following values are recognized by **sudo**:

chroot=string

The root directory to use when running the command.

closefrom=number

If specified, **sudo** will close all files descriptors with a value of *number* or higher.

command=string

Fully qualified path to the command to be executed.

cwd=string

The current working directory to change to when executing the command. If **sudo** is unable to change to the new working directory, the command will not be run unless *cwd_optional* is also set (see below).

cwd_optional=bool

If enabled, **sudo** will treat an inability to change to the new working directory as a non-fatal error. This setting has no effect unless *cwd* is also set.

exec_background=bool

By default, **sudo** runs a command as the foreground process as long as **sudo** itself is running in the foreground. When *exec_background* is enabled and the command is being run in a pseudo-terminal (due to I/O logging or the *use_pty* setting), the command will be run as a background process. Attempts to read from the controlling terminal (or to change terminal settings) will result in the command being suspended with the SIGTTIN signal (or SIGTTOU in the case of terminal settings). If this happens when **sudo** is a foreground process, the command will be granted the controlling terminal and resumed in the foreground with no user intervention required. The advantage of initially running the command in the background is that **sudo** need not read from the terminal unless the command explicitly requests it. Otherwise, any terminal input must be passed to the command, whether it has required it or not (the kernel buffers terminals so it is not possible to tell whether the command really wants the input). This is different from historic *sudo* behavior or when the command is not being run in a pseudo-terminal.

For this to work seamlessly, the operating system must support the automatic restarting of system calls. Unfortunately, not all operating systems do this by default, and even those that do may have bugs. For example, macOS fails to restart the **tcgetattr()** and **tcsetattr()** system calls (this is a bug in macOS). Furthermore, because this behavior depends on the command stopping with the SIGTTIN or SIGTTOU signals, programs that catch these signals and suspend themselves with a different signal (usually SIGTOP) will not be automatically foregrounded. Some versions of the linux **su(1)** command behave this way. Because of this, a plugin should not set *exec_background* unless it is explicitly enabled by the administrator and there should be a way to enable or disable it on a per-command basis.

This setting has no effect unless I/O logging is enabled or *use_pty* is enabled.

execfd=number

If specified, **sudo** will use the **fexecve(2)** system call to execute the command instead of **execve(2)**. The specified *number* must refer to an open file descriptor.

iolog_compress=bool

Set to true if the I/O logging plugins, if any, should compress the log data. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_group=string

The group that will own newly created I/O log files and directories. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_mode=octal

The file permission mode to use when creating I/O log files and directories. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_user=string

The user that will own newly created I/O log files and directories. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_path=string

Fully qualified path to the file or directory in which I/O log is to be stored. This is a hint to the I/O logging plugin which may choose to ignore it. If no I/O logging plugin is loaded, this setting has no effect.

iolog_stdin=bool

Set to true if the I/O logging plugins, if any, should log the standard input if it is not connected to a terminal device. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_stdout=bool

Set to true if the I/O logging plugins, if any, should log the standard output if it is not connected to a terminal device. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_stderr=bool

Set to true if the I/O logging plugins, if any, should log the standard error if it is not connected to a terminal device. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_ttyin=bool

Set to true if the I/O logging plugins, if any, should log all terminal input. This only includes input typed by the user and not from a pipe or redirected from a file. This is a hint to the I/O logging plugin which may choose to ignore it.

iolog_ttyout=bool

Set to true if the I/O logging plugins, if any, should log all terminal output. This only includes output to the screen, not output to a pipe or file. This is a hint to the I/O logging plugin which may choose to ignore it.

login_class=string

BSD login class to use when setting resource limits and nice value (optional). This option is only set on systems that support login classes.

nice=int

Nice value (priority) to use when executing the command. The nice value, if specified, overrides the priority associated with the *login_class* on BSD systems.

noexec=bool

If set, prevent the command from executing other programs.

preserve_fds=list

A comma-separated list of file descriptors that should be preserved, regardless of the value of the *closefrom* setting. Only available starting with API version 1.5.

preserve_groups=bool

If set, **sudo** will preserve the user's group vector instead of initializing the group vector based on *runas_user*.

rlimit_as=soft,hard

The maximum size to which the process's address space may grow (in bytes), if supported by the operating system. The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of "infinity" indicates that there is no limit. A value of "user" will cause the invoking user's resource limit to be preserved. A value of "default" will cause the target user's default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_core=soft,hard

The largest size core dump file that may be created (in bytes). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_cpu=soft,hard

The maximum amount of CPU time that the process may use (in seconds). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_data=soft,hard

The maximum size of the data segment for the process (in bytes). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_fsize=soft,hard

The largest size file that the process may create (in bytes). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_locks=soft,hard

The maximum number of locks that the process may establish, if supported by the operating system. The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_memlock=soft,hard

The maximum size that the process may lock in memory (in bytes), if supported by the operating system. The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_nofile=soft,hard

The maximum number of files that the process may have open. The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the tar-

get user's default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_nproc=soft,hard

The maximum number of processes that the user may run simultaneously. The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_rss=soft,hard

The maximum size to which the process’s resident set size may grow (in bytes). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

rlimit_stack=soft,hard

The maximum size to which the process’s stack may grow (in bytes). The soft and hard limits are separated by a comma. If only a single value is specified, both the hard and soft limits are set. A value of “infinity” indicates that there is no limit. A value of “user” will cause the invoking user’s resource limit to be preserved. A value of “default” will cause the target user’s default resource limit to be used on systems that allow per-user resource limits to be configured. Only available starting with API version 1.17.

runas_egid=gid

Effective group-ID to run the command as. If not specified, the value of *runas_gid* is used.

runas_euid=uid

Effective user-ID to run the command as. If not specified, the value of *runas_uid* is used.

runas_gid=gid

Group-ID to run the command as.

runas_group=string

The name of the group the command will run as, if it is different from the *runas_user*’s default group. This value is provided for auditing purposes only, the **sudo** front-end uses *runas_egid* and *runas_gid* when executing the command.

runas_groups=list

The supplementary group vector to use for the command in the form of a comma-separated list of group-IDs. If *preserve_groups* is set, this option is ignored.

runas_uid=uid

User-ID to run the command as.

runas_user=string

The name of the user the command will run as, which should correspond to *runas_euid* (or *runas_uid* if *runas_euid* is not set). This value is provided for auditing purposes only, the **sudo** front-end uses *runas_euid* and *runas_uid* when executing the command.

selinux_role=string
SELinux role to use when executing the command.

selinux_type=string
SELinux type to use when executing the command.

set_utmp=bool
Create a utmp (or utmpx) entry when a pseudo-terminal is allocated. By default, the new entry will be a copy of the user's existing utmp entry (if any), with the tty, time, type, and pid fields updated.

sudoedit=bool
Set to true when in *sudoedit* mode. The plugin may enable *sudoedit* mode even if **sudo** was not invoked as **sudoedit**. This allows the plugin to perform command substitution and transparently enable *sudoedit* when the user attempts to run an editor.

sudoedit_checkdir=bool
Set to false to disable directory writability checks in **sudoedit**. By default, **sudoedit** 1.8.16 and higher will check all directory components of the path to be edited for writability by the invoking user. Symbolic links will not be followed in writable directories and **sudoedit** will refuse to edit a file located in a writable directory. These restrictions are not enforced when **sudoedit** is run by root. The *sudoedit_follow* option can be set to false to disable this check. Only available starting with API version 1.8.

sudoedit_follow=bool
Set to true to allow **sudoedit** to edit files that are symbolic links. By default, **sudoedit** 1.8.15 and higher will refuse to open a symbolic link. The *sudoedit_follow* option can be used to restore the older behavior and allow **sudoedit** to open symbolic links. Only available starting with API version 1.8.

timeout=int
Command timeout. If non-zero then when the timeout expires the command will be killed.

umask=octal
The file creation mask to use when executing the command. This value may be overridden by PAM or login.conf on some systems unless the *umask_override* option is also set.

umask_override=bool
Force the value specified by the *umask* option to override any umask set by PAM or login.conf.

use_pty=bool
Allocate a pseudo-terminal to run the command in, regardless of whether or not I/O logging is in use. By default, **sudo** will only run the command in a pseudo-terminal when an I/O log plugin is loaded.

utmp_user=string
User name to use when constructing a new utmp (or utmpx) entry when *set_utmp* is enabled. This option can be used to set the user field in the utmp entry to the user the command runs as rather than the invoking user. If not set, **sudo** will base the new entry on the invoking user's existing entry.

Unsupported values will be ignored.

argv_out
The NULL-terminated argument vector to pass to the `execve(2)` system call when executing the command. The plugin is responsible for allocating and populating the vector.

user_env_out

The NULL-terminated environment vector to use when executing the command. The plugin is responsible for allocating and populating the vector.

errstr

If the **check_policy()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

list

```
int (*list)(int argc, char * const argv[], int verbose,
           const char *list_user, const char **errstr);
```

List available privileges for the invoking user. Returns 1 on success, 0 on failure, and -1 on error. On error, the plugin may optionally call the **conversation()** or **plugin_printf()** function with **SUDO_CONF_ERROR_MSG** to present additional error information to the user.

Privileges should be output via the **conversation()** or **plugin_printf()** function using **SUDO_CONV_INFO_MSG**.

The function arguments are as follows:

argc The number of elements in *argv*, not counting the final NULL pointer.

argv If non-NUL, an argument vector describing a command the user wishes to check against the policy in the same form as what would be passed to the **execve(2)** system call. If the command is permitted by the policy, the fully-qualified path to the command should be displayed along with any command line arguments.

verbose

Flag indicating whether to list in verbose mode or not.

list_user

The name of a different user to list privileges for if the policy allows it. If NULL, the plugin should list the privileges of the invoking user.

errstr

If the **list()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

validate

```
int (*validate)(const char **errstr);
```

The **validate()** function is called when **sudo** is run with the **-v** option. For policy plugins such as **sudoers** that cache authentication credentials, this function will validate and cache the credentials.

The **validate()** function should be NULL if the plugin does not support credential caching.

Returns 1 on success, 0 on failure, and -1 on error. On error, the plugin may optionally call the **conversation()** or **plugin_printf()** function with SUDO_CONF_ERROR_MSG to present additional error information to the user.

The function arguments are as follows:

errstr

If the **validate()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

invalidate

```
void (*invalidate)(int remove);
```

The **invalidate()** function is called when **sudo** is run with the **-k** or **-K** option. For policy plugins such as **sudoers** that cache authentication credentials, this function will invalidate the credentials. If the *remove* flag is set, the plugin may remove the credentials instead of simply invalidating them.

The **invalidate()** function should be NULL if the plugin does not support credential caching.

init_session

```
int (*init_session)(struct passwd *pwd, char **user_env_out[]);
```

The **init_session()** function is called before **sudo** sets up the execution environment for the command. It is run in the parent **sudo** process and before any user-ID or group-ID changes. This can be used to perform session setup that is not supported by *command_info*, such as opening the PAM session. The **close()** function can be used to tear down the session that was opened by **init_session**.

The *pwd* argument points to a **passwd** struct for the user the command will be run as if the user-ID the command will run as was found in the password database, otherwise it will be NULL.

The *user_env_out* argument points to the environment the command will run in, in the form of a NULL-terminated vector of "name=value" strings. This is the same string passed back to the front-end via the Policy Plugin's *user_env_out* parameter. If the **init_session()** function needs to modify the user environment, it should update the pointer stored in *user_env_out*. The expected use case is to merge the contents of the PAM environment (if any) with the contents of *user_env_out*. NOTE: the *user_env_out* parameter is only available starting with API version 1.2. A plugin **must** check the API version specified by the **sudo** front-end before using *user_env_out*. Failure to do so may result in a crash.

Returns 1 on success, 0 on failure, and -1 on error. On error, the plugin may optionally call the **conversation()** or **plugin_printf()** function with SUDO_CONF_ERROR_MSG to present additional error information to the user.

register_hooks

```
void (*register_hooks)(int version,
                      int (*register_hook)(struct sudo_hook *hook));
```

The **register_hooks()** function is called by the sudo front-end to register any hooks the plugin needs. If the plugin does not support hooks, **register_hooks** should be set to the NULL pointer.

The *version* argument describes the version of the hooks API supported by the **sudo** front-end.

The **register_hook()** function should be used to register any supported hooks the plugin needs. It returns 0 on success, 1 if the hook type is not supported, and -1 if the major version in **struct hook** does not match the front-end's major hook API version.

See the **Hook function API** section below for more information about hooks.

NOTE: the **register_hooks()** function is only available starting with API version 1.2. If the **sudo** front-end doesn't support API version 1.2 or higher, **register_hooks** will not be called.

deregister_hooks

```
void (*deregister_hooks)(int version,
    int (*deregister_hook)(struct sudo_hook *hook));
```

The **deregister_hooks()** function is called by the sudo front-end to deregister any hooks the plugin has registered. If the plugin does not support hooks, **deregister_hooks** should be set to the NULL pointer.

The *version* argument describes the version of the hooks API supported by the **sudo** front-end.

The **deregister_hook()** function should be used to deregister any hooks that were put in place by the **register_hook()** function. If the plugin tries to deregister a hook that the front-end does not support, **deregister_hook** will return an error.

See the **Hook function API** section below for more information about hooks.

NOTE: the **deregister_hooks()** function is only available starting with API version 1.2. If the **sudo** front-end doesn't support API version 1.2 or higher, **deregister_hooks** will not be called.

event_alloc

```
struct sudo_plugin_event * (*event_alloc)(void);
```

The **event_alloc()** function is used to allocate a **struct sudo_plugin_event** which provides access to the main **sudo** event loop. Unlike the other fields, the **event_alloc()** pointer is filled in by the **sudo** front-end, not by the plugin.

See the **Event API** section below for more information about events.

NOTE: the **event_alloc()** function is only available starting with API version 1.15. If the **sudo** front-end doesn't support API version 1.15 or higher, **event_alloc()** will not be set.

errstr

If the **init_session()** function returns a value other than 1, the plugin may store a message describing the failure or error in **errstr**. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in **errstr** must remain valid until the plugin's **close()** function is called.

NOTE: the **errstr** parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using **errstr**. Failure to do so may result in a crash.

Policy Plugin Version Macros

```
/* Plugin API version major/minor. */
#define SUDO_API_VERSION_MAJOR 1
#define SUDO_API_VERSION_MINOR 13
#define SUDO_API_MKVERSION(x, y) ((x << 16) | y)
#define SUDO_API_VERSION SUDO_API_MKVERSION(SUDO_API_VERSION_MAJOR, \
                                         SUDO_API_VERSION_MINOR)
```

```
/* Getters and setters for API version */
#define SUDO_API_VERSION_GET_MAJOR(v) ((v) >> 16)
#define SUDO_API_VERSION_GET_MINOR(v) ((v) & 0xffff)
#define SUDO_API_VERSION_SET_MAJOR(vp, n) do { \
    *(vp) = (*(vp) & 0x0000ffff) | ((n) << 16); \
} while(0)
#define SUDO_API_VERSION_SET_MINOR(vp, n) do { \
    *(vp) = (*(vp) & 0xffff0000) | (n); \
} while(0)
```

I/O plugin API

```
struct io_plugin {
#define SUDO_IO_PLUGIN 2
    unsigned int type; /* always SUDO_IO_PLUGIN */
    unsigned int version; /* always SUDO_API_VERSION */
    int (*open)(unsigned int version, sudo_conv_t conversation,
        sudo_printf_t plugin_printf, char * const settings[],
        char * const user_info[], char * const command_info[],
        int argc, char * const argv[], char * const user_env[],
        char * const plugin_options[], const char **errstr);
    void (*close)(int exit_status, int error); /* wait status or error */
    int (*show_version)(int verbose);
    int (*log_ttyin)(const char *buf, unsigned int len,
        const char **errstr);
    int (*log_ttyout)(const char *buf, unsigned int len,
        const char **errstr);
    int (*log_stdin)(const char *buf, unsigned int len,
        const char **errstr);
    int (*log_stdout)(const char *buf, unsigned int len,
        const char **errstr);
    int (*log_stderr)(const char *buf, unsigned int len,
        const char **errstr);
    void (*register_hooks)(int version,
        int (*register_hook)(struct sudo_hook *hook));
    void (*deregister_hooks)(int version,
        int (*deregister_hook)(struct sudo_hook *hook));
    int (*change_winsize)(unsigned int lines, unsigned int cols,
        const char **errstr);
    int (*log_suspend)(int signo, const char **errstr);
    struct sudo_plugin_event * (*event_alloc)(void);
};
```

When an I/O plugin is loaded, **sudo** runs the command in a pseudo-terminal. This makes it possible to log the input and output from the user's session. If any of the standard input, standard output, or standard error do not correspond to a tty, **sudo** will open a pipe to capture the I/O for logging before passing it on.

The **log_ttyin** function receives the raw user input from the terminal device (note that this will include input even when echo is disabled, such as when a password is read). The **log_ttyout** function receives output from the pseudo-terminal that is suitable for replaying the user's session at a later time. The **log_stdin()**, **log_stdout()**, and **log_stderr()** functions are only called if the standard input, standard output, or standard error respectively correspond to something other than a tty.

Any of the logging functions may be set to the NULL pointer if no logging is to be performed. If the open function returns 0, no I/O will be sent to the plugin.

If a logging function returns an error (-1), the running command will be terminated and all of the plugin's logging functions will be disabled. Other I/O logging plugins will still receive any remaining input or output that has not yet been processed.

If an input logging function rejects the data by returning 0, the command will be terminated and the data will not be passed to the command, though it will still be sent to any other I/O logging plugins. If an output logging function rejects the data by returning 0, the command will be terminated and the data will not be written to the terminal, though it will still be sent to any other I/O logging plugins.

The audit_plugin struct has the following fields:

type The type field should always be set to SUDO_IO_PLUGIN.

version

The version field should be set to SUDO_API_VERSION.

This allows sudo to determine the API version the plugin was built against.

open

```
int (*open)(unsigned int version, sudo_conv_t conversation,
            sudo_printf_t plugin_printf, char * const settings[],
            char * const user_info[], char * const command_info[],
            int argc, char * const argv[], char * const user_env[],
            char * const plugin_options[]);
```

The open() function is run before the log_ttyin(), log_ttyout(), log_stdin(), log_stdout(), log_stderr(), log_suspend(), change_winsize(), or show_version() functions are called. It is only called if the version is being requested or if the policy plugin's check_policy() function has returned successfully. It returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error. In the latter case, sudo will print a usage message before it exits. If an error occurs, the plugin may optionally call the conversation() or plugin_printf() function with SUDO_CONF_ERROR_MSG to present additional error information to the user.

The function arguments are as follows:

version

The version passed in by sudo allows the plugin to determine the major and minor version number of the plugin API supported by sudo.

conversation

A pointer to the conversation() function that may be used by the show_version() function to display version information (see show_version() below). The conversation() function may also be used to display additional error message to the user. The conversation() function returns 0 on success and -1 on failure.

plugin_printf

A pointer to a printf()-style function that may be used by the show_version() function to display version information (see show_version below). The plugin_printf() function may also be used to display additional error message to the user. The plugin_printf() function returns number of characters printed on success and -1 on failure.

settings

A vector of user-supplied sudo settings in the form of "name=value" strings. The vector is terminated by a NULL pointer. These settings correspond to options the user specified when run-

ning **sudo**. As such, they will only be present when the corresponding option has been specified on the command line.

When parsing *settings*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible settings.

user_info

A vector of information about the user running the command in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *user_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

command_info

A vector of information describing the command being run in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *command_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

argc The number of elements in *argv*, not counting the final NULL pointer. It can be zero, when **sudo** is called with **-V**.

argv If non-NUL, an argument vector describing a command the user wishes to run in the same form as what would be passed to the `execve(2)` system call.

user_env

The user’s environment in the form of a NULL-terminated vector of “name=value” strings.

When parsing *user_env*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

plugin_options

Any (non-comment) strings immediately after the plugin path are treated as arguments to the plugin. These arguments are split on a white space boundary and are passed to the plugin in the form of a NULL-terminated array of strings. If no arguments were specified, *plugin_options* will be the NULL pointer.

NOTE: the *plugin_options* parameter is only available starting with API version 1.2. A plugin **must** check the API version specified by the **sudo** front-end before using *plugin_options*. Failure to do so may result in a crash.

errstr

If the `open()` function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin’s `close()` function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

close

```
void (*close)(int exit_status, int error);
```

The **close()** function is called when **sudo** is finished, shortly before it exits.

The function arguments are as follows:

exit_status

The command's exit status, as returned by the `wait(2)` system call, or zero if no command was run. The value of `exit_status` is undefined if `error` is non-zero.

error If the command could not be executed, this is set to the value of `errno` set by the `execve(2)` system call. If the command was successfully executed, the value of `error` is zero.

show_version

```
int (*show_version)(int verbose);
```

The **show_version()** function is called by **sudo** when the user specifies the `-v` option. The plugin may display its version information to the user via the **conversation()** or **plugin_printf()** function using `SUDO_CONV_INFO_MSG`.

Returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error, although the return value is currently ignored.

log_ttyin

```
int (*log_ttyin)(const char *buf, unsigned int len,
                  const char **errstr);
```

The **log_ttyin()** function is called whenever data can be read from the user but before it is passed to the running command. This allows the plugin to reject data if it chooses to (for instance if the input contains banned content). Returns 1 if the data should be passed to the command, 0 if the data is rejected (which will terminate the running command), or -1 if an error occurred.

The function arguments are as follows:

buf The buffer containing user input.

len The length of `buf` in bytes.

errstr

If the **log_ttyin()** function returns a value other than 1, the plugin may store a message describing the failure or error in `errstr`. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in `errstr` must remain valid until the plugin's **close()** function is called.

NOTE: the `errstr` parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using `errstr`. Failure to do so may result in a crash.

log_ttyout

```
int (*log_ttyout)(const char *buf, unsigned int len,
                  const char **errstr);
```

The **log_ttyout()** function is called whenever data can be read from the command but before it is written to the user's terminal. This allows the plugin to reject data if it chooses to (for instance if the output contains banned content). Returns 1 if the data should be passed to the user, 0 if the data is rejected (which will terminate the running command), or -1 if an error occurred.

The function arguments are as follows:

buf The buffer containing command output.

len The length of *buf* in bytes.

errstr

If the **log_ttyout()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

log_stdin

```
int (*log_stdin)(const char *buf, unsigned int len,
                  const char **errstr);
```

The **log_stdin()** function is only used if the standard input does not correspond to a tty device. It is called whenever data can be read from the standard input but before it is passed to the running command. This allows the plugin to reject data if it chooses to (for instance if the input contains banned content). Returns 1 if the data should be passed to the command, 0 if the data is rejected (which will terminate the running command), or -1 if an error occurred.

The function arguments are as follows:

buf The buffer containing user input.

len The length of *buf* in bytes.

errstr

If the **log_stdin()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

log_stdout

```
int (*log_stdout)(const char *buf, unsigned int len,
                  const char **errstr);
```

The **log_stdout()** function is only used if the standard output does not correspond to a tty device. It is called whenever data can be read from the command but before it is written to the standard output. This allows the plugin to reject data if it chooses to (for instance if the output contains banned content). Returns 1 if the data should be passed to the user, 0 if the data is rejected (which will terminate the running command), or -1 if an error occurred.

The function arguments are as follows:

buf The buffer containing command output.

len The length of *buf* in bytes.

errstr

If the **log_stdout()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's

close() function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

```
log_stderr
int (*log_stderr)(const char *buf, unsigned int len,
                  const char **errstr);
```

The **log_stderr()** function is only used if the standard error does not correspond to a tty device. It is called whenever data can be read from the command but before it is written to the standard error. This allows the plugin to reject data if it chooses to (for instance if the output contains banned content). Returns 1 if the data should be passed to the user, 0 if the data is rejected (which will terminate the running command), or -1 if an error occurred.

The function arguments are as follows:

buf The buffer containing command output.

len The length of *buf* in bytes.

errstr

If the **log_stderr()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so may result in a crash.

register_hooks

See the **Policy plugin API** section for a description of **register_hooks**.

deregister_hooks

See the **Policy plugin API** section for a description of **deregister_hooks**.

change_winsize

```
int (*change_winsize)(unsigned int lines, unsigned int cols,
                      const char **errstr);
```

The **change_winsize()** function is called whenever the window size of the terminal changes from the initial values specified in the **user_info** list. Returns -1 if an error occurred, in which case no further calls to **change_winsize()** will be made.

The function arguments are as follows:

lines The number of lines (rows) in the re-sized terminal.

cols The number of columns in the re-sized terminal.

errstr

If the **change_winsize()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

NOTE: the *errstr* parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using *errstr*. Failure to do so

may result in a crash.

```
log_suspend
    int (*log_suspend)(int signo, const char **errstr);
```

The **log_suspend()** function is called whenever a command is suspended or resumed. Logging this information makes it possible to skip the period of time when the command was suspended during playback of a session. Returns -1 if an error occurred, in which case no further calls to **log_suspend()** will be made,

The function arguments are as follows:

signo

The signal that caused the command to be suspended, or SIGCONT if the command was resumed.

errstr

If the **log_suspend()** function returns a value other than 1, the plugin may store a message describing the failure or error in **errstr**. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in **errstr** must remain valid until the plugin's **close()** function is called.

NOTE: the **errstr** parameter is only available starting with API version 1.15. A plugin **must** check the API version specified by the **sudo** front-end before using **errstr**. Failure to do so may result in a crash.

event_alloc

```
    struct sudo_plugin_event * (*event_alloc)(void);
```

The **event_alloc()** function is used to allocate a **struct sudo_plugin_event** which provides access to the main **sudo** event loop. Unlike the other fields, the **event_alloc()** pointer is filled in by the **sudo** front-end, not by the plugin.

See the **Event API** section below for more information about events.

NOTE: the **event_alloc()** function is only available starting with API version 1.15. If the **sudo** front-end doesn't support API version 1.15 or higher, **event_alloc()** will not be set.

I/O Plugin Version Macros

Same as for the **Policy plugin API**.

Audit plugin API

```
/* Audit plugin close function status types. */
#define SUDO_PLUGIN_NO_STATUS          0
#define SUDO_PLUGIN_WAIT_STATUS        1
#define SUDO_PLUGIN_EXEC_ERROR         2
#define SUDO_PLUGIN_SUDO_ERROR         3

#define SUDO_AUDIT_PLUGIN 3
struct audit_plugin {
    unsigned int type; /* always SUDO_AUDIT_PLUGIN */
    unsigned int version; /* always SUDO_API_VERSION */
    int (*open)(unsigned int version, sudo_conv_t conversation,
               sudo_printf_t sudo_printf, char * const settings[],
               char * const user_info[], int submit_optind,
               char * const submit_argv[], char * const submit_envp[],
               char * const plugin_options[], const char **errstr);
```

```

void (*close)(int status_type, int status);
int (*accept)(const char *plugin_name,
              unsigned int plugin_type, char * const command_info[],
              char * const run_argv[], char * const run_envp[],
              const char **errstr);
int (*reject)(const char *plugin_name, unsigned int plugin_type,
              const char *audit_msg, char * const command_info[],
              const char **errstr);
int (*error)(const char *plugin_name, unsigned int plugin_type,
              const char *audit_msg, char * const command_info[],
              const char **errstr);
int (*show_version)(int verbose);
void (*register_hooks)(int version,
                      int (*register_hook)(struct sudo_hook *hook));
void (*deregister_hooks)(int version,
                        int (*deregister_hook)(struct sudo_hook *hook));
struct sudo_plugin_event * (*event_alloc)(void);
}

```

An audit plugin can be used to log successful and unsuccessful attempts to run **sudo** independent of the policy or any I/O plugins. Multiple audit plugins may be specified in **sudo.conf(5)**.

The **audit_plugin** struct has the following fields:

type The **type** field should always be set to **SUDO_AUDIT_PLUGIN**.

version

The **version** field should be set to **SUDO_API_VERSION**.

This allows **sudo** to determine the API version the plugin was built against.

open

```

int (*open)(unsigned int version, sudo_conv_t conversation,
            sudo_printf_t sudo_printf, char * const settings[],
            char * const user_info[], int submit_optind,
            char * const submit_argv[], char * const submit_envp[],
            char * const plugin_options[], const char **errstr);

```

The audit **open()** function is run before any other **sudo** plugin API functions. This makes it possible to audit failures in the other plugins. It returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error. In the latter case, **sudo** will print a usage message before it exits. If an error occurs, the plugin may optionally call the **conversation()** or **plugin_printf()** function with **SUDO_CONF_ERROR_MSG** to present additional error information to the user.

The function arguments are as follows:

version

The version passed in by **sudo** allows the plugin to determine the major and minor version number of the plugin API supported by **sudo**.

conversation

A pointer to the **conversation()** function that may be used by the **show_version()** function to display version information (see **show_version()** below). The **conversation()** function may also be used to display additional error message to the user. The **conversation()** function returns 0 on success, and -1 on failure.

plugin_printf

A pointer to a **printf()**-style function that may be used by the **show_version()** function to display version information (see **show_version** below). The **plugin_printf()** function may also be used to display additional error message to the user. The **plugin_printf()** function returns number of characters printed on success and -1 on failure.

settings

A vector of user-supplied **sudo** settings in the form of “name=value” strings. The vector is terminated by a NULL pointer. These settings correspond to options the user specified when running **sudo**. As such, they will only be present when the corresponding option has been specified on the command line.

When parsing *settings*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible settings.

user_info

A vector of information about the user running the command in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *user_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

submit_optind

The index into *submit_argv* that corresponds to the first entry that is not a command line option. If *submit_argv* only consists of options, which may be the case with the **-l** or **-v** options, *submit_argv[submit_optind]* will evaluate to the NULL pointer.

submit_argv

The argument vector **sudo** was invoked with, including all command line options. The *submit_optind* argument can be used to determine the end of the command line options.

submit_envp

The invoking user’s environment in the form of a NULL-terminated vector of “name=value” strings.

When parsing *submit_envp*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

plugin_options

Any (non-comment) strings immediately after the plugin path are treated as arguments to the plugin. These arguments are split on a white space boundary and are passed to the plugin in the form of a NULL-terminated array of strings. If no arguments were specified, *plugin_options* will be the NULL pointer.

errstr

If the **open()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin’s **close()** function is called.

close

```
void (*close)(int status_type, int status);
```

The **close()** function is called when **sudo** is finished, shortly before it exits.

The function arguments are as follows:

status_type

The type of status being passed. One of SUDO_PLUGIN_NO_STATUS, SUDO_PLUGIN_WAIT_STATUS, SUDO_PLUGIN_EXEC_ERROR or SUDO_PLUGIN_SUDO_ERROR.

status

Depending on the value of *status_type*, this value is either ignored, the command's exit status as returned by the `wait(2)` system call, the value of `errno` set by the `execve(2)` system call, or the value of `errno` resulting from an error in the **sudo** front-end.

accept

```
int (*accept)(const char *plugin_name, unsigned int plugin_type,
char * const command_info[], char * const run_argv[],
char * const run_envp[], const char **errstr);
```

The **accept()** function is called when a command or action is accepted by a policy or approval plugin. The function arguments are as follows:

plugin_name

The name of the plugin that accepted the command or “sudo” for the **sudo** front-end.

plugin_type

The type of plugin that accepted the command, currently either SUDO_POLICY_PLUGIN, SUDO_POLICY_APPROVAL, or SUDO_FRONT_END. The **accept()** function is called multiple times--once for each policy or approval plugin that succeeds and once for the sudo front-end. When called on behalf of the sudo front-end, *command_info* may include information from an I/O logging plugin as well.

Typically, an audit plugin is interested in either the accept status from the **sudo** front-end or from the various policy and approval plugins, but not both. It is possible for the policy plugin to accept a command that is later rejected by an approval plugin, in which case the audit plugin's **accept()** and **reject()** functions will *both* be called.

command_info

An optional vector of information describing the command being run in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *command_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

run_argv

A NULL-terminated argument vector describing a command that will be run in the same form as what would be passed to the `execve(2)` system call.

run_envp

The environment the command will be run with in the form of a NULL-terminated vector of “name=value” strings.

When parsing *run_envp*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

errstr

If the **accept()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

reject

```
int (*reject)(const char *plugin_name, unsigned int plugin_type,
              const char *audit_msg, char * const command_info[],
              const char **errstr);
```

The **reject()** function is called when a command or action is rejected by a plugin. The function arguments are as follows:

plugin_name

The name of the plugin that rejected the command.

plugin_type

The type of plugin that rejected the command, currently either SUDO_POLICY_PLUGIN, SUDO_APPROVAL_PLUGIN, or SUDO_IO_PLUGIN.

Unlike the **accept()** function, the **reject()** function is not called on behalf of the **sudo** front-end.

audit_msg

An optional string describing the reason the command was rejected by the plugin. If the plugin did not provide a reason, *audit_msg* will be the NULL pointer.

command_info

An optional vector of information describing the command being run in the form of "name=value" strings. The vector is terminated by a NULL pointer.

When parsing *command_info*, the plugin should split on the **first** equal sign ('=') since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

errstr

If the **reject()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin's **close()** function is called.

error

```
int (*error)(const char *plugin_name, unsigned int plugin_type,
             const char *audit_msg, char * const command_info[],
             const char **errstr);
```

The **error()** function is called when a plugin or the **sudo** front-end returns an error. The function arguments are as follows:

plugin_name

The name of the plugin that generated the error or "sudo" for the **sudo** front-end.

plugin_type

The type of plugin that generated the error, or SUDO_FRONT_END for the **sudo** front-end.

audit_msg

An optional string describing the plugin error. If the plugin did not provide a description, *audit_msg* will be the NULL pointer.

command_info

An optional vector of information describing the command being run in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing *command_info*, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

errstr

If the **error()** function returns a value other than 1, the plugin may store a message describing the failure or error in *errstr*. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in *errstr* must remain valid until the plugin’s **close()** function is called.

show_version

```
int (*show_version)(int verbose);
```

The **show_version()** function is called by **sudo** when the user specifies the **-V** option. The plugin may display its version information to the user via the **conversation()** or **plugin_printf()** function using **SUDO_CONV_INFO_MSG**. If the user requests detailed version information, the **verbose** flag will be set.

Returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error, although the return value is currently ignored.

register_hooks

See the **Policy plugin API** section for a description of **register_hooks**.

deregister_hooks

See the **Policy plugin API** section for a description of **deregister_hooks**.

event_alloc

```
struct sudo_plugin_event * (*event_alloc)(void);
```

The **event_alloc()** function is used to allocate a **struct sudo_plugin_event** which provides access to the main **sudo** event loop. Unlike the other fields, the **event_alloc()** pointer is filled in by the **sudo** front-end, not by the plugin.

See the **Event API** section below for more information about events.

NOTE: the **event_alloc()** function is only available starting with API version 1.17. If the **sudo** front-end doesn’t support API version 1.17 or higher, **event_alloc()** will not be set.

Approval plugin API

```
struct approval_plugin {
#define SUDO_APPROVAL_PLUGIN 4
    unsigned int type; /* always SUDO_APPROVAL_PLUGIN */
    unsigned int version; /* always SUDO_API_VERSION */
    int (*open)(unsigned int version, sudo_conv_t conversation,
                sudo_printf_t sudo_printf, char * const settings[],
                char * const user_info[], int submit_optind,
                char * const submit_argv[], char * const submit_envp[],
                char * const plugin_options[], const char **errstr);
```

```

void (*close)(void);
int (*check)(char * const command_info[], char * const run_argv[],
             char * const run_envp[], const char **errstr);
int (*show_version)(int verbose);
} ;

```

An approval plugin can be used to apply extra constraints after a command has been accepted by the policy plugin. Unlike the other plugin types, it does not remain open until the command completes. The plugin is opened before a call to `check()` or `show_version()` and closed shortly thereafter (audit plugin functions must be called before the plugin is closed). Multiple approval plugins may be specified in `sudo.conf(5)`.

The `approval_plugin` struct has the following fields:

`type` The `type` field should always be set to `SUDO_APPROVAL_PLUGIN`.

`version`

The `version` field should be set to `SUDO_API_VERSION`.

This allows `sudo` to determine the API version the plugin was built against.

`open`

```

int (*open)(unsigned int version, sudo_conv_t conversation,
            sudo_printf_t sudo_printf, char * const settings[],
            char * const user_info[], int submit_optind,
            char * const submit_argv[], char * const submit_envp[],
            char * const plugin_options[], const char **errstr);

```

The approval `open()` function is run immediately before a call to the plugin's `check()` or `show_version()` functions. It is only called if the version is being requested or if the policy plugin's `check_policy()` function has returned successfully. It returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error. In the latter case, `sudo` will print a usage message before it exits. If an error occurs, the plugin may optionally call the `conversation()` or `plugin_printf()` function with `SUDO_CONF_ERROR_MSG` to present additional error information to the user.

The function arguments are as follows:

`version`

The version passed in by `sudo` allows the plugin to determine the major and minor version number of the plugin API supported by `sudo`.

`conversation`

A pointer to the `conversation()` function that can be used by the plugin to interact with the user (see **Conversation API** for details). Returns 0 on success and -1 on failure.

`plugin_printf`

A pointer to a `printf()`-style function that may be used to display informational or error messages (see **Conversation API** for details). Returns the number of characters printed on success and -1 on failure.

`settings`

A vector of user-supplied `sudo` settings in the form of "name=value" strings. The vector is terminated by a NULL pointer. These settings correspond to options the user specified when running `sudo`. As such, they will only be present when the corresponding option has been specified on the command line.

When parsing `settings`, the plugin should split on the **first** equal sign ('=') since the `name` field will never include one itself but the `value` might.

See the **Policy plugin API** section for a list of all possible settings.

`user_info`

A vector of information about the user running the command in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing `user_info`, the plugin should split on the **first** equal sign (‘=’) since the `name` field will never include one itself but the `value` might.

See the **Policy plugin API** section for a list of all possible strings.

`submit_optind`

The index into `submit_argv` that corresponds to the first entry that is not a command line option. If `submit_argv` only consists of options, which may be the case with the **-l** or **-v** options, `submit_argv[submit_optind]` will evaluate to the NULL pointer.

`submit_argv`

The argument vector **sudo** was invoked with, including all command line options. The `submit_optind` argument can be used to determine the end of the command line options.

`submit_envp`

The invoking user’s environment in the form of a NULL-terminated vector of “name=value” strings.

When parsing `submit_envp`, the plugin should split on the **first** equal sign (‘=’) since the `name` field will never include one itself but the `value` might.

`plugin_options`

Any (non-comment) strings immediately after the plugin path are treated as arguments to the plugin. These arguments are split on a white space boundary and are passed to the plugin in the form of a NULL-terminated array of strings. If no arguments were specified, `plugin_options` will be the NULL pointer.

`errstr`

If the `open()` function returns a value other than 1, the plugin may store a message describing the failure or error in `errstr`. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in `errstr` must remain valid until the plugin’s `close()` function is called.

`close`

```
void (*close)(void);
```

The `close()` function is called after the approval plugin’s `check()` or `show_version()` functions have been called. It takes no arguments. The `close()` function is typically used to perform plugin-specific cleanup, such as the freeing of memory objects allocated by the plugin. If the plugin does not need to perform any cleanup, `close()` may be set to the NULL pointer.

`check`

```
int (*check)(char * const command_info[], char * const run_argv[],
             char * const run_envp[], const char **errstr);
```

The approval `check()` function is run after the policy plugin `check_policy()` function and before any I/O logging plugins. If multiple approval plugins are loaded, they must all succeed for the command to be allowed. It returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error. In the latter case, **sudo** will print a usage message before it exits. If an error occurs, the plugin may optionally call the `conversation()` or `plugin_printf()` function with `SUDO_CONF_ERROR_MSG` to present additional error information to the user.

The function arguments are as follows:

`command_info`

A vector of information describing the command being run in the form of “name=value” strings. The vector is terminated by a NULL pointer.

When parsing `command_info`, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

See the **Policy plugin API** section for a list of all possible strings.

`run_argv`

A NULL-terminated argument vector describing a command that will be run in the same form as what would be passed to the `execve(2)` system call.

`run_envp`

The environment the command will be run with in the form of a NULL-terminated vector of “name=value” strings.

When parsing `run_envp`, the plugin should split on the **first** equal sign (‘=’) since the *name* field will never include one itself but the *value* might.

`errstr`

If the `open()` function returns a value other than 1, the plugin may store a message describing the failure or error in `errstr`. The **sudo** front-end will then pass this value to any registered audit plugins. The string stored in `errstr` must remain valid until the plugin’s `close()` function is called.

`show_version`

```
int (*show_version)(int verbose);
```

The `show_version()` function is called by **sudo** when the user specifies the **-v** option. The plugin may display its version information to the user via the `conversation()` or `plugin_printf()` function using `SUDO_CONV_INFO_MSG`. If the user requests detailed version information, the `verbose` flag will be set.

Returns 1 on success, 0 on failure, -1 if a general error occurred, or -2 if there was a usage error, although the return value is currently ignored.

Signal handlers

The **sudo** front-end installs default signal handlers to trap common signals while the plugin functions are run. The following signals are trapped by default before the command is executed:

- `SIGALRM`
- `SIGHUP`
- `SIGINT`
- `SIGPIPE`
- `SIGQUIT`
- `SIGTERM`
- `SIGTSTP`
- `SIGUSR1`
- `SIGUSR2`

If a fatal signal is received before the command is executed, **sudo** will call the plugin’s `close()` function with an exit status of 128 plus the value of the signal that was received. This allows for consistent logging of commands killed by a signal for plugins that log such information in their `close()` function. An exception to this is `SIGPIPE`, which is ignored until the command is executed.

A plugin may temporarily install its own signal handlers but must restore the original handler before the plugin function returns.

Hook function API

Beginning with plugin API version 1.2, it is possible to install hooks for certain functions called by the **sudo** front-end.

Currently, the only supported hooks relate to the handling of environment variables. Hooks can be used to intercept attempts to get, set, or remove environment variables so that these changes can be reflected in the version of the environment that is used to execute a command. A future version of the API will support hooking internal **sudo** front-end functions as well.

Hook structure

Hooks in **sudo** are described by the following structure:

```
typedef int (*sudo_hook_fn_t)();

struct sudo_hook {
    unsigned int hook_version;
    unsigned int hook_type;
    sudo_hook_fn_t hook_fn;
    void *closure;
};
```

The **sudo_hook** structure has the following fields:

hook_version

The **hook_version** field should be set to **SUDO_HOOK_VERSION**.

hook_type

The **hook_type** field may be one of the following supported hook types:

SUDO_HOOK_SETENV

The C library **setenv(3)** function. Any registered hooks will run before the C library implementation. The **hook_fn** field should be a function that matches the following typedef:

```
typedef int (*sudo_hook_fn_setenv_t)(const char *name,
                                     const char *value, int overwrite, void *closure);
```

If the registered hook does not match the typedef the results are unspecified.

SUDO_HOOK_UNSETENV

The C library **unsetenv(3)** function. Any registered hooks will run before the C library implementation. The **hook_fn** field should be a function that matches the following typedef:

```
typedef int (*sudo_hook_fn_unsetenv_t)(const char *name,
                                       void *closure);
```

SUDO_HOOK_GETENV

The C library **getenv(3)** function. Any registered hooks will run before the C library implementation. The **hook_fn** field should be a function that matches the following typedef:

```
typedef int (*sudo_hook_fn_getenv_t)(const char *name,
                                     char **value, void *closure);
```

If the registered hook does not match the typedef the results are unspecified.

SUDO_HOOK_PUTENV

The C library `putenv(3)` function. Any registered hooks will run before the C library implementation. The `hook_fn` field should be a function that matches the following typedef:

```
typedef int (*sudo_hook_fn_putenv_t)(char *string,
                                     void *closure);
```

If the registered hook does not match the typedef the results are unspecified.

hook_fn

```
sudo_hook_fn_t hook_fn;
```

The `hook_fn` field should be set to the plugin's hook implementation. The actual function arguments will vary depending on the `hook_type` (see `hook_type` above). In all cases, the `closure` field of struct `sudo_hook` is passed as the last function parameter. This can be used to pass arbitrary data to the plugin's hook implementation.

The function return value may be one of the following:

SUDO_HOOK_RET_ERROR

The hook function encountered an error.

SUDO_HOOK_RET_NEXT

The hook completed without error, go on to the next hook (including the system implementation if applicable). For example, a `getenv(3)` hook might return `SUDO_HOOK_RET_NEXT` if the specified variable was not found in the private copy of the environment.

SUDO_HOOK_RET_STOP

The hook completed without error, stop processing hooks for this invocation. This can be used to replace the system implementation. For example, a `setenv` hook that operates on a private copy of the environment but leaves `environ` unchanged.

Note that it is very easy to create an infinite loop when hooking C library functions. For example, a `getenv(3)` hook that calls the `snprintf(3)` function may create a loop if the `snprintf(3)` implementation calls `getenv(3)` to check the locale. To prevent this, you may wish to use a static variable in the hook function to guard against nested calls. For example:

```
static int in_progress = 0; /* avoid recursion */
if (in_progress)
    return SUDO_HOOK_RET_NEXT;
in_progress = 1;
...
in_progress = 0;
return SUDO_HOOK_RET_STOP;
```

Hook API Version Macros

```
/* Hook API version major/minor */
#define SUDO_HOOK_VERSION_MAJOR 1
#define SUDO_HOOK_VERSION_MINOR 0
#define SUDO_HOOK_VERSION SUDO_API_MKVERSION(SUDO_HOOK_VERSION_MAJOR, \
                                             SUDO_HOOK_VERSION_MINOR)
```

For getters and setters see the **Policy plugin API**.

Event API

When `sudo` runs a command, it uses an event loop to service signals and I/O. Events may be triggered based on time, a file or socket descriptor becoming ready, or due to receipt of a signal. Starting with API

version 1.15, it is possible for a plugin to participate in this event loop by calling the **event_alloc()** function.

Event structure

Events are described by the following structure:

```
typedef void (*sudo_plugin_ev_callback_t)(int fd, int what, void *closure);

struct sudo_plugin_event {
    int (*set)(struct sudo_plugin_event *pev, int fd, int events,
               sudo_plugin_ev_callback_t callback, void *closure);
    int (*add)(struct sudo_plugin_event *pev, struct timespec *timeout);
    int (*del)(struct sudo_plugin_event *pev);
    int (*pending)(struct sudo_plugin_event *pev, int events,
                   struct timespec *ts);
    int (*fd)(struct sudo_plugin_event *pev);
    void (*setbase)(struct sudo_plugin_event *pev, void *base);
    void (*loopbreak)(struct sudo_plugin_event *pev);
    void (*free)(struct sudo_plugin_event *pev);
};
```

The sudo_plugin_event struct contains the following function pointers:

```
set()
    int (*set)(struct sudo_plugin_event *pev, int fd, int events,
               sudo_plugin_ev_callback_t callback, void *closure);
```

The **set()** function takes the following arguments:

struct sudo_plugin_event *pev
A pointer to the struct sudo_plugin_event itself.

fd The file or socket descriptor for I/O-based events or the signal number for signal events. For time-based events, **fd** must be -1.

events

The following values determine what will trigger the event callback:

SUDO_PLUGIN_EV_TIMEOUT

callback is run after the specified timeout expires

SUDO_PLUGIN_EV_READ

callback is run when the file descriptor is readable

SUDO_PLUGIN_EV_WRITE

callback is run when the file descriptor is writable

SUDO_PLUGIN_EV_PERSIST

event is persistent and remains enabled until explicitly deleted

SUDO_PLUGIN_EV_SIGNAL

callback is run when the specified signal is received

The SUDO_PLUGIN_EV_PERSIST flag may be ORed with any of the event types. It is also possible to OR SUDO_PLUGIN_EV_READ and SUDO_PLUGIN_EV_WRITE together to run the callback when a descriptor is ready to be either read from or written to. All other event values are mutually exclusive.

```
sudo_plugin_ev_callback_t callback
    typedef void (*sudo_plugin_ev_callback_t)(int fd, int what,
        void *closure);
```

The function to call when an event is triggered. The **callback()** function is run with the following arguments:

fd The file or socket descriptor for I/O-based events or the signal number for signal events.

what

The event type that triggered that callback. For events that have multiple event types (for example SUDO_PLUGIN_EV_READ and SUDO_PLUGIN_EV_WRITE) or have an associated timeout, *what* can be used to determine why the callback was run.

closure

The generic pointer that was specified in the **set()** function.

closure

A generic pointer that will be passed to the callback function.

The **set()** function returns 1 on success, and -1 if a error occurred.

add()

```
int (*add)(struct sudo_plugin_event *pev, struct timespec *timeout);
```

The **add()** function adds the event *pev* to **sudo**'s event loop. The event must have previously been initialized via the **set()** function. If the *timeout* argument is not NULL, it should specify a (relative) timeout after which the event will be triggered if the main event criteria has not been met. This is often used to implement an I/O timeout where the event will fire if a descriptor is not ready within a certain time period. If the event is already present in the event loop, its *timeout* will be adjusted to match the new value, if any.

The **add()** function returns 1 on success, and -1 if a error occurred.

del()

```
int (*del)(struct sudo_plugin_event *pev);
```

The **del()** function deletes the event *pev* from **sudo**'s event loop. Deleted events can be added back via the **add()** function.

The **del()** function returns 1 on success, and -1 if a error occurred.

pending()

```
int (*pending)(struct sudo_plugin_event *pev, int events,
    struct timespec *ts);
```

The **pending()** function can be used to determine whether one or more events is pending. The *events* argument specifies which events to check for. See the **set()** function for a list of valid event types. If SUDO_PLUGIN_EV_TIMEOUT is specified in *events*, the event has an associated timeout and the *ts* pointer is non-NUL, it will be filled in with the remaining time.

fd()

```
int (*fd)(struct sudo_plugin_event *pev);
```

The **fd()** function returns the descriptor or signal number associated with the event *pev*.

setbase()

```
void (*setbase)(struct sudo_plugin_event *pev, void *base);
```

The **setbase()** function sets the underlying event *base* for *pev* to the specified value. This can be used to move an event created via **event_alloc()** to a new event loop allocated by sudo's event subsystem. If *base* is NULL, *pev*'s event base is reset to the default value, which corresponds to **sudo**'s main event loop. Using this function requires linking the plugin with the sudo_util library. It is unlikely to be used outside of the **sudoers** plugin.

```
loopbreak()
void (*loopbreak)(struct sudo_plugin_event *pev);
```

The **loopbreak()** function causes **sudo**'s event loop to exit immediately and the running command to be terminated.

```
free()
void (*free)(struct sudo_plugin_event *pev);
```

The **free()** function deletes the event *pev* from the event loop and frees the memory associated with it.

Remote command execution

The **sudo** front-end does not support running remote commands. However, starting with **sudo** 1.8.8, the **-h** option may be used to specify a remote host that is passed to the policy plugin. A plugin may also accept a *runas_user* in the form of “user@hostname” which will work with older versions of **sudo**. It is anticipated that remote commands will be supported by executing a “helper” program. The policy plugin should setup the execution environment such that the **sudo** front-end will run the helper which, in turn, will connect to the remote host and run the command.

For example, the policy plugin could utilize **ssh** to perform remote command execution. The helper program would be responsible for running **ssh** with the proper options to use a private key or certificate that the remote host will accept and run a program on the remote host that would setup the execution environment accordingly.

Note that remote **sudoedit** functionality must be handled by the policy plugin, not **sudo** itself as the front-end has no knowledge that a remote command is being executed. This may be addressed in a future revision of the plugin API.

Conversation API

If the plugin needs to interact with the user, it may do so via the **conversation()** function. A plugin should not attempt to read directly from the standard input or the user's tty (neither of which are guaranteed to exist). The caller must include a trailing newline in *msg* if one is to be printed.

A **printf()**-style function is also available that can be used to display informational or error messages to the user, which is usually more convenient for simple messages where no user input is required.

Conversation function structures

The conversation function takes as arguments pointers to the following structures:

```
struct sudo_conv_message {
#define SUDO_CONV_PROMPT_ECHO_OFF 0x0001 /* do not echo user input */
#define SUDO_CONV_PROMPT_ECHO_ON 0x0002 /* echo user input */
#define SUDO_CONV_ERROR_MSG 0x0003 /* error message */
#define SUDO_CONV_INFO_MSG 0x0004 /* informational message */
#define SUDO_CONV_PROMPT_MASK 0x0005 /* mask user input */
#define SUDO_CONV_PROMPT_ECHO_OK 0x1000 /* flag: allow echo if no tty */
#define SUDO_CONV_PREFER_TTY 0x2000 /* flag: use tty if possible */

    int msg_type;
    int timeout;
```

```

        const char *msg;
};

#define SUDO_CONV REPL_MAX      1023

struct sudo_conv_reply {
    char *reply;
};

typedef int (*sudo_conv_callback_fn_t)(int signo, void *closure);
struct sudo_conv_callback {
    unsigned int version;
    void *closure;
    sudo_conv_callback_fn_t on_suspend;
    sudo_conv_callback_fn_t on_resume;
};

```

Pointers to the **conversation()** and **printf()**-style functions are passed in to the plugin's **open()** function when the plugin is initialized. The following type definitions can be used in the declaration of the **open()** function:

```

typedef int (*sudo_conv_t)(int num_msgs,
    const struct sudo_conv_message msgs[],
    struct sudo_conv_reply replies[], struct sudo_conv_callback *callback);

typedef int (*sudo_printf_t)(int msg_type, const char *fmt, ...);

```

To use the **conversation()** function, the plugin must pass an array of **sudo_conv_message** and **sudo_conv_reply** structures. There must be a **struct sudo_conv_message** and **struct sudo_conv_reply** for each message in the conversation, that is, both arrays must have the same number of elements. Each **struct sudo_conv_reply** must have its *reply* member initialized to NULL. The **struct sudo_conv_callbackpointer**, if not NULL, should contain function pointers to be called when the **sudo** process is suspended and/or resumed during conversation input. The *on_suspend* and *on_resume* functions are called with the signal that caused **sudo** to be suspended and the *closure* pointer from the **struct sudo_conv_callback**. These functions should return 0 on success and -1 on error. On error, the conversation will end and the conversation function will return a value of -1. The intended use is to allow the plugin to release resources, such as locks, that should not be held indefinitely while suspended and then reacquire them when the process is resumed. Note that the functions are not actually invoked from within a signal handler.

The *msg_type* must be set to one of the following values:

SUDO_CONV_PROMPT_ECHO_OFF

Prompt the user for input with echo disabled; this is generally used for passwords. The reply will be stored in the *replies* array, and it will never be NULL.

SUDO_CONV_PROMPT_ECHO_ON

Prompt the user for input with echo enabled. The reply will be stored in the *replies* array, and it will never be NULL.

SUDO_CONV_ERROR_MSG

Display an error message. The message is written to the standard error unless the **SUDO_CONV_PREFER_TTY** flag is set, in which case it is written to the user's terminal if possible.

SUDO_CONV_INFO_MSG

Display a message. The message is written to the standard output unless the `SUDO_CONV_PREFER_TTY` flag is set, in which case it is written to the user's terminal if possible.

SUDO_CONV_PROMPT_MASK

Prompt the user for input but echo an asterisk character for each character read. The reply will be stored in the `replies` array, and it will never be `NULL`. This can be used to provide visual feedback to the user while reading sensitive information that should not be displayed.

In addition to the above values, the following flag bits may also be set:

SUDO_CONV_PROMPT_ECHO_OK

Allow input to be read when echo cannot be disabled when the message type is `SUDO_CONV_PROMPT_ECHO_OFF` or `SUDO_CONV_PROMPT_MASK`. By default, `sudo` will refuse to read input if the echo cannot be disabled for those message types.

SUDO_CONV_PREFER_TTY

When displaying a message via `SUDO_CONV_ERROR_MSG` or `SUDO_CONV_INFO_MSG`, try to write the message to the user's terminal. If the terminal is unavailable, the standard error or standard output will be used, depending upon whether `SUDO_CONV_ERROR_MSG` or `SUDO_CONV_INFO_MSG` was used. The user's terminal is always used when possible for input, this flag is only used for output.

The `timeout` in seconds until the prompt will wait for no more input. A zero value implies an infinite timeout.

The plugin is responsible for freeing the reply buffer located in each `struct sudo_conv_reply`, if it is not `NULL`. `SUDO_CONV REPL_MAX` represents the maximum length of the reply buffer (not including the trailing NUL character). In practical terms, this is the longest password `sudo` will support.

The `printf()`-style function uses the same underlying mechanism as the `conversation()` function but only supports `SUDO_CONV_INFO_MSG` and `SUDO_CONV_ERROR_MSG` for the `msg_type` parameter. It can be more convenient than using the `conversation()` function if no user reply is needed and supports standard `printf()` escape sequences.

See the sample plugin for an example of the `conversation()` function usage.

Plugin invocation order

As of `sudo` 1.9.0, the plugin `open()` and `close()` functions are called in the following order:

1. audit open
2. policy open
3. approval open
4. approval close
5. I/O log open
6. command runs
7. command exits
8. I/O log close
9. policy close
10. audit close

11. sudo exits

Prior to **sudo** 1.9.0, the I/O log **close()** function was called *after* the policy **close()** function.

Sudoers group plugin API

The **sudoers** plugin supports its own plugin interface to allow non-Unix group lookups. This can be used to query a group source other than the standard Unix group database. Two sample group plugins are bundled with **sudo**, *group_file*, and *system_group*, are detailed in *sudoers(5)*. Third party group plugins include a QAS AD plugin available from Quest Software.

A group plugin must declare and populate a *sudoers_group_plugin* struct in the global scope. This structure contains pointers to the functions that implement plugin initialization, cleanup, and group lookup.

```
struct sudoers_group_plugin {
    unsigned int version;
    int (*init)(int version, sudo_printf_t sudo_printf,
                char *const argv[]);
    void (*cleanup)(void);
    int (*query)(const char *user, const char *group,
                 const struct passwd *pwd);
};
```

The *sudoers_group_plugin* struct has the following fields:

version

The *version* field should be set to **GROUP_API_VERSION**.

This allows **sudoers** to determine the API version the group plugin was built against.

init

```
int (*init)(int version, sudo_printf_t plugin_printf,
            char *const argv[]);
```

The **init()** function is called after *sudoers* has been parsed but before any policy checks. It returns 1 on success, 0 on failure (or if the plugin is not configured), and -1 if an error occurred. If an error occurs, the plugin may call the **plugin_printf()** function with **SUDO_CONF_ERROR_MSG** to present additional error information to the user.

The function arguments are as follows:

version

The *version* passed in by **sudoers** allows the plugin to determine the major and minor version number of the group plugin API supported by **sudoers**.

plugin_printf

A pointer to a **printf()**-style function that may be used to display informational or error message to the user. Returns the number of characters printed on success and -1 on failure.

argv A NULL-terminated array of arguments generated from the *group_plugin* option in *sudoers*. If no arguments were given, *argv* will be NULL.

cleanup

```
void (*cleanup)();
```

The **cleanup()** function is called when **sudoers** has finished its group checks. The plugin should free any memory it has allocated and close open file handles.

query

```
int (*query)(const char *user, const char *group,
             const struct passwd *pwd);
```

The **query()** function is used to ask the group plugin whether *user* is a member of *group*.

The function arguments are as follows:

user The name of the user being looked up in the external group database.

group

The name of the group being queried.

pwd The password database entry for *user*, if any. If *user* is not present in the password database, *pwd* will be NULL.

Group API Version Macros

```
/* Sudoers group plugin version major/minor */
#define GROUP_API_VERSION_MAJOR 1
#define GROUP_API_VERSION_MINOR 0
#define GROUP_API_VERSION ((GROUP_API_VERSION_MAJOR << 16) | \
                         GROUP_API_VERSION_MINOR)
```

For getters and setters see the **Policy plugin API**.

PLUGIN API CHANGELOG

The following revisions have been made to the Sudo Plugin API.

Version 1.0

Initial API version.

Version 1.1 (sudo 1.8.0)

The I/O logging plugin's **open()** function was modified to take the `command_info` list as an argument.

Version 1.2 (sudo 1.8.5)

The Policy and I/O logging plugins' **open()** functions are now passed a list of plugin parameters if any are specified in `sudo.conf(5)`.

A simple hooks API has been introduced to allow plugins to hook in to the system's environment handling functions.

The `init_session` Policy plugin function is now passed a pointer to the user environment which can be updated as needed. This can be used to merge in environment variables stored in the PAM handle before a command is run.

Version 1.3 (sudo 1.8.7)

Support for the `exec_background` entry has been added to the `command_info` list.

The `max_groups` and `plugin_dir` entries were added to the `settings` list.

The `version()` and `close()` functions are now optional. Previously, a missing `version()` or `close()` function would result in a crash. If no policy plugin `close()` function is defined, a default `close()` function will be provided by the **sudo** front-end that displays a warning if the command could not be executed.

The **sudo** front-end now installs default signal handlers to trap common signals while the plugin functions are run.

Version 1.4 (sudo 1.8.8)

The *remote_host* entry was added to the *settings* list.

Version 1.5 (sudo 1.8.9)

The *preserve_fds* entry was added to the *command_info* list.

Version 1.6 (sudo 1.8.11)

The behavior when an I/O logging plugin returns an error (-1) has changed. Previously, the **sudo** front-end took no action when the **log_ttyin()**, **log_ttyout()**, **log_stdin()**, **log_stdout()**, or **log_stderr()** function returned an error.

The behavior when an I/O logging plugin returns 0 has changed. Previously, output from the command would be displayed to the terminal even if an output logging function returned 0.

Version 1.7 (sudo 1.8.12)

The *plugin_path* entry was added to the *settings* list.

The *debug_flags* entry now starts with a debug file path name and may occur multiple times if there are multiple plugin-specific Debug lines in the **sudo.conf(5)** file.

Version 1.8 (sudo 1.8.15)

The *sudoedit_checkdir* and *sudoedit_follow* entries were added to the *command_info* list. The default value of *sudoedit_checkdir* was changed to true in sudo 1.8.16.

The sudo *conversation* function now takes a pointer to a *struct sudo_conv_callback* as its fourth argument. The *sudo_conv_t* definition has been updated to match. The plugin must specify that it supports plugin API version 1.8 or higher to receive a conversation function pointer that supports this argument.

Version 1.9 (sudo 1.8.16)

The *execfd* entry was added to the *command_info* list.

Version 1.10 (sudo 1.8.19)

The *umask* entry was added to the *user_info* list. The *iolog_group*, *iolog_mode*, and *iolog_user* entries were added to the *command_info* list.

Version 1.11 (sudo 1.8.20)

The *timeout* entry was added to the *settings* list.

Version 1.12 (sudo 1.8.21)

The *change_winsize* field was added to the *io_plugin* struct.

Version 1.13 (sudo 1.8.26)

The *log_suspend* field was added to the *io_plugin* struct.

Version 1.14 (sudo 1.8.29)

The *umask_override* entry was added to the *command_info* list.

Version 1.15 (sudo 1.9.0)

The *cwd_optional* entry was added to the *command_info* list.

The *event_alloc* field was added to the *policy_plugin* and *io_plugin* structs.

The *errstr* argument was added to the policy and I/O plugin functions which the plugin function can use to return an error string. This string may be used by the audit plugin to report failure or error conditions set by the other plugins.

The **close()** function is now called regardless of whether or not a command was actually executed. This makes it possible for plugins to perform cleanup even when a command was not run.

SUDO_CONV_REPL_MAX has increased from 255 to 1023 bytes.

Support for audit and approval plugins was added.

Version 1.16 (sudo 1.9.3)

Initial resource limit values were added to the user_info list.

The *cmnd_chroot* and *cmnd_cwd* entries were added to the settings list.

Version 1.17 (sudo 1.9.4)

The *event_alloc* field was added to the audit_plugin and approval_plugin structs.

Version 1.18 (sudo 1.9.9)

The policy may now set resource limit values in the command_info list.

SEE ALSO

`sudo.conf(5)`, `sudoers(5)`, `sudo(8)`

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo_root – How to run administrative commands

SYNOPSIS

sudo *command*

sudo -i

INTRODUCTION

By default, the password for the user "root" (the system administrator) is locked. This means you cannot log in as root or use su. Instead, the installer will set up sudo to allow the user that is created during install to run all administrative commands.

This means that in the terminal you can use sudo for commands that require root privileges. All programs in the menu will use a graphical sudo to prompt for a password. When sudo asks for a password, it needs **your password**, this means that a root password is not needed.

To run a command which requires root privileges in a terminal, simply prepend **sudo** in front of it. To get an interactive root shell, use **sudo -i**.

ALLOWING OTHER USERS TO RUN SUDO

By default, only the user who installed the system is permitted to run sudo. To add more administrators, i. e. users who can run sudo, you have to add these users to the group 'sudo' by doing one of the following steps:

- * In a shell, do

sudo adduser *username* sudo

- * Use the graphical "Users & Groups" program in the "System settings" menu to add the new user to the **sudo** group.

BENEFITS OF USING SUDO

The benefits of leaving root disabled by default include the following:

- * Users do not have to remember an extra password, which they are likely to forget.
- * The installer is able to ask fewer questions.
- * It avoids the "I can do anything" interactive login by default – you will be prompted for a password before major changes can happen, which should make you think about the consequences of what you are doing.
- * Sudo adds a log entry of the command(s) run (in **/var/log/auth.log**).
- * Every attacker trying to brute-force their way into your box will know it has an account named root and will try that first. What they do not know is what the usernames of your other users are.
- * Allows easy transfer for admin rights, in a short term or long term period, by adding and removing users from the sudo group, while not compromising the root account.
- * sudo can be set up with a much more fine-grained security policy.
- * On systems with more than one administrator using sudo avoids sharing a password amongst them.

DOWNSIDES OF USING SUDO

Although for desktops the benefits of using sudo are great, there are possible issues which need to be noted:

- * Redirecting the output of commands run with sudo can be confusing at first. For instance consider

sudo ls > /root/somefile

will not work since it is the shell that tries to write to that file. You can use

ls | sudo tee /root/somefile

to get the behaviour you want.

- * In a lot of office environments the ONLY local user on a system is root. All other users are imported using NSS techniques such as nss-ldap. To setup a workstation, or fix it, in the case of a network failure where nss-ldap is broken, root is required. This tends to leave the system unusable. An extra local user, or an enabled root password is needed here.

GOING BACK TO A TRADITIONAL ROOT ACCOUNT

This is not recommended!

To enable the root account (i.e. set a password) use:

sudo passwd root

Afterwards, edit the sudo configuration with **sudo visudo** and comment out the line

%sudo ALL=(ALL) ALL

to disable sudo access to members of the sudo group.

SEE ALSO

sudo(8), <https://wiki.ubuntu.com/RootSudo>

NAME

sudo_sendlog — send sudo I/O log to log server

SYNOPSIS

```
sudo_sendlog [ -AnV ] [ -b ca_bundle ] [ -c cert_file ] [ -h host ] [ -i iolog-id ]
[ -k key_file ] [ -p port ] [ -r restart-point ] [ -R reject-reason ]
[ -s stop-point ] [ -t number ] path
```

DESCRIPTION

sudo_sendlog can be used to send the existing **sudoers** I/O log *path* to a remote log server such as **sudo_logsvrd(8)** for central storage.

The options are as follows:

-A, --accept-only

Only send the accept event, not the I/O associated with the log. This can be used to test the logging of accept events without any associated I/O.

-b, --ca-bundle

The path to a certificate authority bundle file, in PEM format, to use instead of the system's default certificate authority database when authenticating the log server. The default is to use the system's default certificate authority database.

-c, --cert

The path to the client's certificate file in PEM format. This setting is required when the connection to the remote log server is secured with TLS.

--help

Display a short help message to the standard output and exit.

-h, --host

Connect to the specified *host* instead of localhost.

-i, --iolog-id

Use the specified *iolog-id* when restarting a log transfer. The *iolog-id* is reported by the server when it creates the remote I/O log. This option may only be used in conjunction with the **-r** option.

-k, --key

The path to the client's private key file in PEM format. This setting is required when the connection to the remote log server is secured with TLS.

-n, --no-verify

If specified, the server's certificate will not be verified during the TLS handshake. By default, **sudo_sendlog** verifies that the server's certificate is valid and that it contains either the server's host name or its IP address. This setting is only supported when the connection to the remote log server is secured with TLS.

-p, --port

Use the specified network *port* when connecting to the log server instead of the default, port 30344.

-r, --restart

Restart an interrupted connection to the log server. The specified *restart-point* is used to tell the server the point in time at which to continue the log. The *restart-point* is specified in the form "seconds,nanoseconds" and is usually the last commit point received from the server. The **-i** option must also be specified when restarting a transfer .

-R, --reject

Send a reject event for the command using the specified *reject-reason*, even though it was actually accepted locally. This can be used to test the logging of reject events; no I/O will be sent.

-s, --stop-after

Stop sending log records and close the connection when *stop-point* is reached. This can be used for testing purposes to send a partial I/O log to the server. Partial logs can be restarted using the **-r** option. The *stop-point* is an elapsed time specified in the form “seconds,nanoseconds”.

-t, --test

Open *number* simultaneous connections to the log server and send the specified I/O log file on each one. This option is useful for performance testing.

-V, --version

Print the **sudo_sendlog** version and exit.

Debugging sendlog

sudo_sendlog supports a flexible debugging framework that is configured via Debug lines in the sudo.conf(5) file.

For more information on configuring sudo .conf(5), please refer to its manual.

FILES

/etc/sudo.conf Sudo front-end configuration

SEE ALSO

`sudo.conf(5)`, `sudo(8)`, `sudo_logsrvd(8)`

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the **CONTRIBUTORS** file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in `sudo_sendlog`, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo_sendlog is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudo, sudoedit — execute a command as another user

SYNOPSIS

```
sudo -h | -K | -k | -v
sudo -v[ -ABknS] [-g group] [-h host] [-p prompt] [-u user]
sudo -l[ -ABknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
sudo [ -ABbEHnPS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
      [-R directory] [-r role] [-t type] [-T timeout] [-u user] [VAR=value]
      [-i | -s] [command]
sudoedit [ -ABknS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
      [-R directory] [-r role] [-t type] [-T timeout] [-u user] file ...
```

DESCRIPTION

sudo allows a permitted user to execute a *command* as the superuser or another user, as specified by the security policy. The invoking user's real (*not effective*) user-ID is used to determine the user name with which to query the security policy.

sudo supports a plugin architecture for security policies, auditing, and input/output logging. Third parties can develop and distribute their own plugins to work seamlessly with the **sudo** front-end. The default security policy is *sudoers*, which is configured via the file /etc/sudoers, or via LDAP. See the **Plugins** section for more information.

The security policy determines what privileges, if any, a user has to run **sudo**. The policy may require that users authenticate themselves with a password or another authentication mechanism. If authentication is required, **sudo** will exit if the user's password is not entered within a configurable time limit. This limit is policy-specific; the default password prompt timeout for the *sudoers* security policy is 0 minutes.

Security policies may support credential caching to allow the user to run **sudo** again for a period of time without requiring authentication. By default, the *sudoers* policy caches credentials on a per-terminal basis for 15 minutes. See the *timestamp_type* and *timestamp_timeout* options in *sudoers*(5) for more information. By running **sudo** with the **-v** option, a user can update the cached credentials without running a *command*.

On systems where **sudo** is the primary method of gaining superuser privileges, it is imperative to avoid syntax errors in the security policy configuration files. For the default security policy, *sudoers*(5), changes to the configuration files should be made using the *visudo*(8) utility which will ensure that no syntax errors are introduced.

When invoked as **sudoedit**, the **-e** option (described below), is implied.

Security policies and audit plugins may log successful and failed attempts to run **sudo**. If an I/O plugin is configured, the running command's input and output may be logged as well.

The options are as follows:

-A, --askpass

Normally, if **sudo** requires a password, it will read it from the user's terminal. If the **-A** (*askpass*) option is specified, a (possibly graphical) helper program is executed to read the user's password and output the password to the standard output. If the **SUDO_ASKPASS** environment variable is set, it specifies the path to the helper program. Otherwise, if *sudo.conf*(5) contains a line specifying the *askpass* program, that value will be used. For example:

```
# Path to askpass helper program
Path askpass /usr/X11R6/bin/ssh-askpass
```

If no askpass program is available, **sudo** will exit with an error.

-B, --bell

Ring the bell as part of the password prompt when a terminal is present. This option has no effect if an askpass program is used.

-b, --background

Run the given command in the background. Note that it is not possible to use shell job control to manipulate background processes started by **sudo**. Most interactive commands will fail to work properly in background mode.

-C num, --close-from=num

Close all file descriptors greater than or equal to *num* before executing a command. Values less than three are not permitted. By default, **sudo** will close all open file descriptors other than standard input, standard output, and standard error when executing a command. The security policy may restrict the user's ability to use this option. The *sudoers* policy only permits use of the **-C** option when the administrator has enabled the *closefrom_override* option.

-D directory, --chdir=directory

Run the command in the specified *directory* instead of the current working directory. The security policy may return an error if the user does not have permission to specify the working directory.

-E, --preserve-env

Indicates to the security policy that the user wishes to preserve their existing environment variables. The security policy may return an error if the user does not have permission to preserve the environment.

--preserve-env=list

Indicates to the security policy that the user wishes to add the comma-separated list of environment variables to those preserved from the user's environment. The security policy may return an error if the user does not have permission to preserve the environment. This option may be specified multiple times.

-e, --edit

Edit one or more files instead of running a command. In lieu of a path name, the string "sudoeedit" is used when consulting the security policy. If the user is authorized by the policy, the following steps are taken:

1. Temporary copies are made of the files to be edited with the owner set to the invoking user.
2. The editor specified by the policy is run to edit the temporary files. The *sudoers* policy uses the *SUDO_EDITOR*, *VISUAL* and *EDITOR* environment variables (in that order). If none of *SUDO_EDITOR*, *VISUAL* or *EDITOR* are set, the first program listed in the *editor sudoers(5)* option is used.
3. If they have been modified, the temporary files are copied back to their original location and the temporary versions are removed.

To help prevent the editing of unauthorized files, the following restrictions are enforced unless explicitly allowed by the security policy:

- Symbolic links may not be edited (version 1.8.15 and higher).
- Symbolic links along the path to be edited are not followed when the parent directory is writable by the invoking user unless that user is root (version 1.8.16 and higher).

- Files located in a directory that is writable by the invoking user may not be edited unless that user is root (version 1.8.16 and higher).

Users are never allowed to edit device special files.

If the specified file does not exist, it will be created. Note that unlike most commands run by **sudo**, the editor is run with the invoking user's environment unmodified. If the temporary file becomes empty after editing, the user will be prompted before it is installed. If, for some reason, **sudo** is unable to update a file with its edited version, the user will receive a warning and the edited copy will remain in a temporary file.

-g group, --group=group

Run the command with the primary group set to *group* instead of the primary group specified by the target user's password database entry. The *group* may be either a group name or a numeric group-ID (GID) prefixed with the '#' character (e.g., #0 for GID 0). When running a command as a GID, many shells require that the '#' be escaped with a backslash ('\'). If no **-u** option is specified, the command will be run as the invoking user. In either case, the primary group will be set to *group*. The *sudoer*'s policy permits any of the target user's groups to be specified via the **-g** option as long as the **-P** option is not in use.

-H, --set-home

Request that the security policy set the HOME environment variable to the home directory specified by the target user's password database entry. Depending on the policy, this may be the default behavior.

-h, --help

Display a short help message to the standard output and exit.

-h host, --host=host

Run the command on the specified *host* if the security policy plugin supports remote commands. Note that the *sudoer*'s plugin does not currently support running remote commands. This may also be used in conjunction with the **-l** option to list a user's privileges for the remote host.

-i, --login

Run the shell specified by the target user's password database entry as a login shell. This means that login-specific resource files such as *.profile*, *.bash_profile*, or *.login* will be read by the shell. If a command is specified, it is passed to the shell as a simple command using the **-c** option. The command and any arguments are concatenated, separated by spaces, after escaping each character (including white space) with a backslash ('\') except for alphanumerics, underscores, hyphens, and dollar signs. If no command is specified, an interactive shell is executed. **sudo** attempts to change to that user's home directory before running the shell. The command is run with an environment similar to the one a user would receive at log in. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details. The *Command environment* section in the *sudoers(5)* manual documents how the **-i** option affects the environment in which a command is run when the *sudoers* policy is in use.

-K, --remove-timestamp

Similar to the **-k** option, except that it removes the user's cached credentials entirely and may not be used in conjunction with a command or other option. This option does not require a password. Not all security policies support credential caching.

-k, --reset-timestamp

When used without a command, invalidates the user's cached credentials. In other words, the next time **sudo** is run a password will be required. This option does not require a password,

and was added to allow a user to revoke **sudo** permissions from a .logout file.

When used in conjunction with a command or an option that may require a password, this option will cause **sudo** to ignore the user's cached credentials. As a result, **sudo** will prompt for a password (if one is required by the security policy) and will not update the user's cached credentials.

Not all security policies support credential caching.

-l, --list

If no *command* is specified, list the allowed (and forbidden) commands for the invoking user (or the user specified by the **-U** option) on the current host. A longer list format is used if this option is specified multiple times and the security policy supports a verbose output format.

If a *command* is specified and is permitted by the security policy, the fully-qualified path to the command is displayed along with any command line arguments. If a *command* is specified but not allowed by the policy, **sudo** will exit with a status value of 1.

-n, --non-interactive

Avoid prompting the user for input of any kind. If a password is required for the command to run, **sudo** will display an error message and exit.

-P, --preserve-groups

Preserve the invoking user's group vector unaltered. By default, the *sudoers* policy will initialize the group vector to the list of groups the target user is a member of. The real and effective group-IDs, however, are still set to match the target user.

-p *prompt*, --prompt=*prompt*

Use a custom password prompt with optional escape sequences. The following percent ('%') escape sequences are supported by the *sudoers* policy:

%H

expanded to the host name including the domain name (only if the machine's host name is fully qualified or the *fqdn* option is set in *sudoers*(5))

%h

expanded to the local host name without the domain name

%p

expanded to the name of the user whose password is being requested (respects the *rootpw*, *targetpw*, and *runaspw* flags in *sudoers*(5))

%U

expanded to the login name of the user the command will be run as (defaults to root unless the **-u** option is also specified)

%u

expanded to the invoking user's login name

%%

two consecutive '%' characters are collapsed into a single '%' character

The custom prompt will override the default prompt specified by either the security policy or the *SUDO_PROMPT* environment variable. On systems that use PAM, the custom prompt will also override the prompt specified by a PAM module unless the *passprompt_override* flag is disabled in *sudoers*.

-R directory, --chroot=directory

Change to the specified root *directory* (see *chroot(8)*) before running the command. The security policy may return an error if the user does not have permission to specify the root directory.

-r role, --role=role

Run the command with an SELinux security context that includes the specified *role*.

-s, --stdin

Write the prompt to the standard error and read the password from the standard input instead of using the terminal device.

-s, --shell

Run the shell specified by the SHELL environment variable if it is set or the shell specified by the invoking user's password database entry. If a command is specified, it is passed to the shell as a simple command using the **-c** option. The command and any arguments are concatenated, separated by spaces, after escaping each character (including white space) with a backslash ('\\') except for alphanumerics, underscores, hyphens, and dollar signs. If no command is specified, an interactive shell is executed. Note that most shells behave differently when a command is specified as compared to an interactive session; consult the shell's manual for details.

-t type, --type=type

Run the command with an SELinux security context that includes the specified *type*. If no *type* is specified, the default type is derived from the role.

-U user, --other-user=user

Used in conjunction with the **-l** option to list the privileges for *user* instead of for the invoking user. The security policy may restrict listing other users' privileges. The *sudoer*'s policy only allows root or a user with the ALL privilege on the current host to use this option.

-T timeout, --command-timeout=timeout

Used to set a timeout for the command. If the timeout expires before the command has exited, the command will be terminated. The security policy may restrict the ability to set command timeouts. The *sudoer*'s policy requires that user-specified timeouts be explicitly enabled.

-u user, --user=user

Run the command as a user other than the default target user (usually *root*). The *user* may be either a user name or a numeric user-ID (UID) prefixed with the '#' character (e.g.,#0 for UID 0). When running commands as a UID, many shells require that the '#' be escaped with a backslash ('\\'). Some security policies may restrict UIDs to those listed in the password database. The *sudoer*'s policy allows UIDs that are not in the password database as long as the *targetpw* option is not set. Other security policies may not support this.

-v, --version

Print the **sudo** version string as well as the version string of any configured plugins. If the invoking user is already root, the **-V** option will display the arguments passed to configure when **sudo** was built; plugins may display additional information such as default options.

-v, --validate

Update the user's cached credentials, authenticating the user if necessary. For the *sudoers* plugin, this extends the **sudo** timeout for another 15 minutes by default, but does not run a command. Not all security policies support cached credentials.

-- The **--** option indicates that **sudo** should stop processing command line arguments.

Options that take a value may only be specified once unless otherwise indicated in the description. This is to help guard against problems caused by poorly written scripts that invoke **sudo** with user-controlled input.

Environment variables to be set for the command may also be passed on the command line in the form of `VAR=value`, e.g., `LD_LIBRARY_PATH=/usr/local/pkg/lib`. Variables passed on the command line are subject to restrictions imposed by the security policy plugin. The *sudoers* policy subjects variables passed on the command line to the same restrictions as normal environment variables with one important exception. If the `setenv v` option is set in *sudoers*, the command to be run has the `SETENV` tag set or the command matched is `ALL`, the user may set variables that would otherwise be forbidden. See *sudoers*(5) for more information.

COMMAND EXECUTION

When **sudo** executes a command, the security policy specifies the execution environment for the command. Typically, the real and effective user and group and IDs are set to match those of the target user, as specified in the password database, and the group vector is initialized based on the group database (unless the `-P` option was specified).

The following parameters may be specified by security policy:

- real and effective user-ID
- real and effective group-ID
- supplementary group-IDs
- the environment list
- current working directory
- file creation mode mask (umask)
- SELinux role and type
- scheduling priority (aka nice value)

Process model

There are two distinct ways **sudo** can run a command.

If an I/O logging plugin is configured or if the security policy explicitly requests it, a new pseudo-terminal (“pty”) is allocated and `fork(2)` is used to create a second **sudo** process, referred to as the *monitor*. The *monitor* creates a new terminal session with itself as the leader and the pty as its controlling terminal, calls `fork(2)`, sets up the execution environment as described above, and then uses the `execve(2)` system call to run the command in the child process. The *monitor* exists to relay job control signals between the user’s existing terminal and the pty the command is being run in. This makes it possible to suspend and resume the command. Without the monitor, the command would be in what POSIX terms an “orphaned process group” and it would not receive any job control signals from the kernel. When the command exits or is terminated by a signal, the *monitor* passes the command’s exit status to the main **sudo** process and exits. After receiving the command’s exit status, the main **sudo** passes the command’s exit status to the security policy’s close function and exits.

If no pty is used, **sudo** calls `fork(2)`, sets up the execution environment as described above, and uses the `execve(2)` system call to run the command in the child process. The main **sudo** process waits until the command has completed, then passes the command’s exit status to the security policy’s close function and exits. As a special case, if the policy plugin does not define a close function, **sudo** will execute the command directly instead of calling `fork(2)` first. The *sudoers* policy plugin will only define a close function when I/O logging is enabled, a pty is required, an SELinux role is specified, the command has an associated timeout, or the `pam_session` or `pam_setcred` options are enabled. Note that `pam_session` and `pam_setcred` are enabled by default on systems using PAM.

On systems that use PAM, the security policy's close function is responsible for closing the PAM session. It may also log the command's exit status.

Signal handling

When the command is run as a child of the **sudo** process, **sudo** will relay signals it receives to the command. The SIGINT and SIGQUIT signals are only relayed when the command is being run in a new pty or when the signal was sent by a user process, not the kernel. This prevents the command from receiving SIGINT twice each time the user enters control-C. Some signals, such as SIGSTOP and SIGKILL, cannot be caught and thus will not be relayed to the command. As a general rule, SIGTSTP should be used instead of SIGSTOP when you wish to suspend a command being run by **sudo**.

As a special case, **sudo** will not relay signals that were sent by the command it is running. This prevents the command from accidentally killing itself. On some systems, the **reboot(8)** command sends SIGTERM to all non-system processes other than itself before rebooting the system. This prevents **sudo** from relaying the SIGTERM signal it received back to **reboot(8)**, which might then exit before the system was actually rebooted, leaving it in a half-dead state similar to single user mode. Note, however, that this check only applies to the command run by **sudo** and not any other processes that the command may create. As a result, running a script that calls **reboot(8)** or **shutdown(8)** via **sudo** may cause the system to end up in this undefined state unless the **reboot(8)** or **shutdown(8)** are run using the **exec()** family of functions instead of **system()** (which interposes a shell between the command and the calling process).

If no I/O logging plugins are loaded and the policy plugin has not defined a **close()** function, set a command timeout, or required that the command be run in a new pty, **sudo** may execute the command directly instead of running it as a child process.

Plugins

Plugins may be specified via Plugin directives in the **sudo.conf(5)** file. They may be loaded as dynamic shared objects (on systems that support them), or compiled directly into the **sudo** binary. If no **sudo.conf(5)** file is present, or if it doesn't contain any Plugin lines, **sudo** will use **sudoers(5)** for the policy, auditing, and I/O logging plugins. See the **sudo.conf(5)** manual for details of the **/etc/sudo.conf** file and the **sudo_plugin(5)** manual for more information about the **sudo** plugin architecture.

EXIT VALUE

Upon successful execution of a command, the exit status from **sudo** will be the exit status of the program that was executed. If the command terminated due to receipt of a signal, **sudo** will send itself the same signal that terminated the command.

If the **-l** option was specified without a command, **sudo** will exit with a value of 0 if the user is allowed to run **sudo** and they authenticated successfully (as required by the security policy). If a command is specified with the **-l** option, the exit value will only be 0 if the command is permitted by the security policy, otherwise it will be 1.

If there is an authentication failure, a configuration/permission problem, or if the given command cannot be executed, **sudo** exits with a value of 1. In the latter case, the error string is printed to the standard error. If **sudo** cannot stat(2) one or more entries in the user's PATH, an error is printed to the standard error. (If the directory does not exist or if it is not really a directory, the entry is ignored and no error is printed.) This should not happen under normal circumstances. The most common reason for stat(2) to return "permission denied" is if you are running an automounter and one of the directories in your PATH is on a machine that is currently unreachable.

SECURITY NOTES

sudo tries to be safe when executing external commands.

To prevent command spoofing, **sudo** checks "." and "" (both denoting current directory) last when searching for a command in the user's PATH (if one or both are in the PATH). Depending on the security policy, the user's PATH environment variable may be modified, replaced, or passed unchanged to the program that **sudo** executes.

Users should *never* be granted **sudo** privileges to execute files that are writable by the user or that reside in a directory that is writable by the user. If the user can modify or replace the command there is no way to limit what additional commands they can run.

Please note that **sudo** will normally only log the command it explicitly runs. If a user runs a command such as **sudo su** or **sudo sh**, subsequent commands run from that shell are not subject to **sudo**'s security policy. The same is true for commands that offer shell escapes (including most editors). If I/O logging is enabled, subsequent commands will have their input and/or output logged, but there will not be traditional logs for those commands. Because of this, care must be taken when giving users access to commands via **sudo** to verify that the command does not inadvertently give the user an effective root shell. For information on ways to address this, please see the *Preventing shell escapes* section in **sudoers(5)**.

To prevent the disclosure of potentially sensitive information, **sudo** disables core dumps by default while it is executing (they are re-enabled for the command that is run). This historical practice dates from a time when most operating systems allowed set-user-ID processes to dump core by default. To aid in debugging **sudo** crashes, you may wish to re-enable core dumps by setting "disable_coredump" to false in the **sudo.conf(5)** file as follows:

```
Set disable_coredump false
```

See the **sudo.conf(5)** manual for more information.

ENVIRONMENT

sudo utilizes the following environment variables. The security policy has control over the actual content of the command's environment.

EDITOR	Default editor to use in -e (sudoedit) mode if neither SUDO_EDITOR nor VISUAL is set.
MAIL	Set to the mail spool of the target user when the -i option is specified, or when env_reset is enabled in sudoers (unless MAIL is present in the env_keep list).
HOME	Set to the home directory of the target user when the -i or -H options are specified, when the -s option is specified and set_home is set in sudoers , when always_set_home is enabled in sudoers , or when env_reset is enabled in sudoers and HOME is not present in the env_keep list.
LOGNAME	Set to the login name of the target user when the -i option is specified, when the set_logname option is enabled in sudoers , or when the env_reset option is enabled in sudoers (unless LOGNAME is present in the env_keep list).
PATH	May be overridden by the security policy.
SHELL	Used to determine shell to run with -s option.
SUDO_ASKPASS	Specifies the path to a helper program used to read the password if no terminal is available or if the -A option is specified.
SUDO_COMMAND	Set to the command run by sudo, including command line arguments. The command line arguments are truncated at 4096 characters to prevent a potential execution error.

SUDO_EDITOR	Default editor to use in -e (sudoedit) mode.
SUDO_GID	Set to the group-ID of the user who invoked sudo.
SUDO_PROMPT	Used as the default password prompt unless the -p option was specified.
SUDO_PS1	If set, PS1 will be set to its value for the program being run.
SUDO_UID	Set to the user-ID of the user who invoked sudo.
SUDO_USER	Set to the login name of the user who invoked sudo.
USER	Set to the same value as LOGNAME, described above.
VISUAL	Default editor to use in -e (sudoedit) mode if SUDO_EDITOR is not set.

FILES

/etc/sudo.conf	sudo front-end configuration
----------------	-------------------------------------

EXAMPLES

Note: the following examples assume a properly configured security policy.

To get a file listing of an unreadable directory:

```
$ sudo ls /usr/local/protected
```

To list the home directory of user yaz on a machine where the file system holding ~yaz is not exported as root:

```
$ sudo -u yaz ls ~yaz
```

To edit the index.html file as user www:

```
$ sudoedit -u www ~www/htdocs/index.html
```

To view system logs only accessible to root and users in the adm group:

```
$ sudo -g adm more /var/log/syslog
```

To run an editor as jim with a different primary group:

```
$ sudoedit -u jim -g audio ~jim/sound.txt
```

To shut down a machine:

```
$ sudo shutdown -r +15 "quick reboot"
```

To make a usage listing of the directories in the /home partition. Note that this runs the commands in a sub-shell to make the cd and file redirection work.

```
$ sudo sh -c "cd /home ; du -s * | sort -rn > USAGE"
```

DIAGNOSTICS

Error messages produced by **sudo** include:

editing files in a writable directory is not permitted

By default, **sudoedit** does not permit editing a file when any of the parent directories are writable by the invoking user. This avoids a race condition that could allow the user to overwrite an arbitrary file. See the *sudoedit_c heckdir* option in sudoers(5) for more information.

editing symbolic links is not permitted

By default, **sudoedit** does not follow symbolic links when opening files. See the *sudoedit_follow* option in sudoers(5) for more information.

effective uid is not 0, is sudo installed setuid root?

sudo was not run with root privileges. The **sudo** binary must be owned by the root user and have the set-user-ID bit set. Also, it must not be located on a file system mounted with the ‘nosuid’ option or on an NFS file system that maps uid 0 to an unprivileged uid.

effective uid is not 0, is sudo on a file system with the ‘nosuid’ option set or an NFS file system without root privileges?

sudo was not run with root privileges. The **sudo** binary has the proper owner and permissions but it still did not run with root privileges. The most common reason for this is that the file system the **sudo** binary is located on is mounted with the ‘nosuid’ option or it is an NFS file system that maps uid 0 to an unprivileged uid.

fatal error, unable to load plugins

An error occurred while loading or initializing the plugins specified in **sudo.conf(5)**.

invalid environment variable name

One or more environment variable names specified via the **-E** option contained an equal sign (‘=’). The arguments to the **-E** option should be environment variable names without an associated value.

no password was provided

When **sudo** tried to read the password, it did not receive any characters. This may happen if no terminal is available (or the **-s** option is specified) and the standard input has been redirected from **/dev/null**.

a terminal is required to read the password

sudo needs to read the password but there is no mechanism available for it to do so. A terminal is not present to read the password from, **sudo** has not been configured to read from the standard input, the **-s** option was not used, and no askpass helper has been specified either via the **sudo.conf(5)** file or the **SUDO_ASKPASS** environment variable.

no writable temporary directory found

sudoedit was unable to find a usable temporary directory in which to store its intermediate files.

The “no new privileges” flag is set, which prevents sudo from running as root.

sudo was run by a process that has the Linux “no new privileges” flag is set. This causes the set-user-ID bit to be ignored when running an executable, which will prevent **sudo** from functioning. The most likely cause for this is running **sudo** within a container that sets this flag. Check the documentation to see if it is possible to configure the container such that the flag is not set.

sudo must be owned by uid 0 and have the setuid bit set

sudo was not run with root privileges. The **sudo** binary does not have the correct owner or permissions. It must be owned by the root user and have the set-user-ID bit set.

sudoedit is not supported on this platform

It is only possible to run **sudoedit** on systems that support setting the effective user-ID.

timed out reading password

The user did not enter a password before the password timeout (5 minutes by default) expired.

you do not exist in the passwd database

Your user-ID does not appear in the system passwd database.

you may not specify environment variables in edit mode

It is only possible to specify environment variables when running a command. When editing a file, the editor is run with the user’s environment unmodified.

SEE ALSO

`su(1)`, `stat(2)`, `login_cap(3)`, `passwd(5)`, `sudo.conf(5)`, `sudo_plugin(5)`, `sudoers(5)`,
`sudoers_timestamp(5)`, `sudoreplay(8)`, `visudo(8)`

HISTORY

See the HISTORY file in the `sudo` distribution (<https://www.sudo.ws/history.html>) for a brief history of `sudo`.

AUTHORS

Many people have worked on `sudo` over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the `sudo` distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to `sudo`.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if that user is allowed to run arbitrary commands via `sudo`. Also, many programs (such as editors) allow the user to run commands via shell escapes, thus avoiding `sudo`'s checks. However, on most systems it is possible to prevent shell escapes with the `sudoers(5)` plugin's `noexec` functionality.

It is not meaningful to run the `cd` command directly via `sudo`, e.g.,

```
$ sudo cd /usr/local/protected
```

since when the command exits the parent process (your shell) will still be the same. Please see the **EXAMPLES** section for more information.

Running shell scripts via `sudo` can expose the same kernel bugs that make set-user-ID shell scripts unsafe on some operating systems (if your OS has a `/dev/fd/` directory, set-user-ID shell scripts are generally safe).

BUGS

If you feel you have found a bug in `sudo`, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the `sudo-users` mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

`sudo` is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with `sudo` or <https://www.sudo.ws/license.html> for complete details.

NAME

sudoers — default sudo security policy plugin

DESCRIPTION

The **sudoers** policy plugin determines a user's **sudo** privileges. It is the default **sudo** policy plugin. The policy is driven by the /etc/sudoers file or, optionally, in LDAP. The policy format is described in detail in the **SUDOERS FILE FORMAT** section. For information on storing **sudoers** policy information in LDAP, please see sudoers.ldap(5).

Configuring sudo.conf for sudoers

sudo consults the sudo.conf(5) file to determine which plugins to load. If no sudo.conf(5) file is present, or if it contains no Plugin lines, **sudoers** will be used for auditing, policy decisions and I/O logging. To explicitly configure sudo.conf(5) to use the **sudoers** plugin, the following configuration can be used.

```
Plugin sudoers_audit sudoers.so
Plugin sudoers_policy sudoers.so
Plugin sudoers_io sudoers.so
```

Starting with **sudo** 1.8.5, it is possible to specify optional arguments to the **sudoers** plugin in the sudo.conf(5) file. Plugin arguments, if any, should be listed after the path to the plugin (i.e., after sudoers.so). The arguments are only effective for the plugin that opens (and parses) the sudoers file.

For **sudo** version 1.9.1 and higher, this is the *sudoers_audit* plugin. For older versions, it is the *sudoers_policy* plugin. Multiple arguments may be specified, separated by white space. For example:

```
Plugin sudoers_audit sudoers.so sudoers_mode=0400 error_recovery=false
```

The following plugin arguments are supported:

error_recovery=bool

The *error_recovery* argument can be used to control whether **sudoers** should attempt to recover from syntax errors in the sudoers file. If set to *true* (the default), **sudoers** will try to recover from a syntax error by discarding the portion of the line that contains the error until the end of the line. A value of *false* will disable error recovery. Prior to version 1.9.3, no error recovery was performed.

ldap_conf=pathname

The *ldap_conf* argument can be used to override the default path to the ldap.conf file.

ldap_secret=pathname

The *ldap_secret* argument can be used to override the default path to the ldap.secret file.

sudoers_file=pathname

The *sudoers_file* argument can be used to override the default path to the sudoers file.

sudoers_uid=user-ID

The *sudoers_uid* argument can be used to override the default owner of the sudoers file. It should be specified as a numeric user-ID.

sudoers_gid=group-ID

The *sudoers_gid* argument can be used to override the default group of the sudoers file. It must be specified as a numeric group-ID (not a group name).

sudoers_mode=mode

The *sudoers_mode* argument can be used to override the default file mode for the sudoers file. It should be specified as an octal value.

For more information on configuring `sudo.conf(5)`, please refer to its manual.

User Authentication

The **`sudoers`** security policy requires that most users authenticate themselves before they can use **`sudo`**. A password is not required if the invoking user is root, if the target user is the same as the invoking user, or if the policy has disabled authentication for the user or command. Unlike **`su(1)`**, when **`sudoers`** requires authentication, it validates the invoking user's credentials, not the target user's (or root's) credentials. This can be changed via the *rootpw*, *targetpw* and *runaspw* flags, described later.

If a user who is not listed in the policy tries to run a command via **`sudo`**, mail is sent to the proper authorities. The address used for such mail is configurable via the *mailto* Defaults entry (described later) and defaults to `root`.

Note that no mail will be sent if an unauthorized user tries to run **`sudo`** with the **`-l`** or **`-v`** option unless there is an authentication error and either the *mail_always* or *mail_badpass* flags are enabled. This allows users to determine for themselves whether or not they are allowed to use **`sudo`**. By default, all attempts to run **`sudo`** (successful or not) are logged, regardless of whether or not mail is sent.

If **`sudo`** is run by root and the `SUDO_USER` environment variable is set, the **`sudoers`** policy will use this value to determine who the actual user is. This can be used by a user to log commands through `sudo` even when a root shell has been invoked. It also allows the **`-e`** option to remain useful even when invoked via a sudo-run script or program. Note, however, that the *sudoers* file lookup is still done for root, not the user specified by `SUDO_USER`.

`sudoers` uses per-user time stamp files for credential caching. Once a user has been authenticated, a record is written containing the user-ID that was used to authenticate, the terminal session ID, the start time of the session leader (or parent process) and a time stamp (using a monotonic clock if one is available). The user may then use **`sudo`** without a password for a short period of time (15 minutes unless overridden by the *timestamp_timeout* option). By default, **`sudoers`** uses a separate record for each terminal, which means that a user's login sessions are authenticated separately. The *timestamp_type* option can be used to select the type of time stamp record **`sudoers`** will use.

Logging

By default, **`sudoers`** logs both successful and unsuccessful attempts (as well as errors). The *log_allowed* and *log_denied* flags can be used to control this behavior. Messages can be logged to `syslog(3)`, a log file, or both. The default is to log to `syslog(3)` but this is configurable via the `syslog` and `logfile` settings. See **LOG FORMAT** for a description of the log file format.

`sudoers` is also capable of running a command in a pseudo-terminal and logging all input and/or output. The standard input, standard output, and standard error can be logged even when not associated with a terminal. I/O logging is not on by default but can be enabled using the *log_input* and *log_output* options as well as the `LOG_INPUT` and `LOG_OUTPUT` command tags. See **I/O LOG FILES** for details on how I/O log files are stored.

Starting with version 1.9, the *log_servers* setting may be used to send event and I/O log data to a remote server running **`sudo_logsrvd`** or another service that implements the protocol described by `sudo_logsrv.proto(5)`.

Command environment

Since environment variables can influence program behavior, **`sudoers`** provides a means to restrict which variables from the user's environment are inherited by the command to be run. There are two distinct ways **`sudoers`** can deal with environment variables.

By default, the *env_reset* flag is enabled. This causes commands to be executed with a new, minimal environment. On AIX (and Linux systems without PAM), the environment is initialized with the contents of the

/etc/environment file. The HOME, MAIL, SHELL, LOGNAME and USER environment variables are initialized based on the target user and the SUDO_* variables are set based on the invoking user. Additional variables, such as DISPLAY, PATH and TERM, are preserved from the invoking user's environment if permitted by the *env_check*, or *env_keep* options. A few environment variables are treated specially. If the PATH and TERM variables are not preserved from the user's environment, they will be set to default values. The LOGNAME and USER are handled as a single entity. If one of them is preserved (or removed) from the user's environment, the other will be as well. If LOGNAME and USER are to be preserved but only one of them is present in the user's environment, the other will be set to the same value. This avoids an inconsistent environment where one of the variables describing the user name is set to the invoking user and one is set to the target user. Environment variables with a value beginning with () are removed unless both the name and value parts are matched by *env_keep* or *env_check*, as they may be interpreted as functions by the **bash** shell. Prior to version 1.8.11, such variables were always removed.

If, however, the *env_reset* flag is disabled, any variables not explicitly denied by the *env_check* and *env_delete* options are allowed and their values are inherited from the invoking process. Prior to version 1.8.21, environment variables with a value beginning with () were always removed. Beginning with version 1.8.21, a pattern in *env_delete* is used to match **bash** shell functions instead. Since it is not possible to block all potentially dangerous environment variables, use of the default *env_reset* behavior is encouraged.

Environment variables specified by *env_check*, *env_delete*, or *env_keep* may include one or more '*' characters which will match zero or more characters. No other wildcard characters are supported.

By default, environment variables are matched by name. However, if the pattern includes an equal sign ('='), both the variables name and value must match. For example, a **bash** shell function could be matched as follows:

```
env_keep += "BASH_FUNC_my_func%%=( )*"
```

Without the "=()*" suffix, this would not match, as **bash** shell functions are not preserved by default.

The complete list of environment variables that are preserved or removed, as modified by global Defaults parameters in *sudoers*, is displayed when **sudo** is run by root with the **-V** option. Please note that the list of environment variables to remove varies based on the operating system **sudo** is running on.

Other **sudoers** options may influence the command environment, such as *always_set_home*, *secure_path*, *set_logname*, and *set_home*.

On systems that support PAM where the **pam_env** module is enabled for **sudo**, variables in the PAM environment may be merged in to the environment. If a variable in the PAM environment is already present in the user's environment, the value will only be overridden if the variable was not preserved by **sudoers**. When *env_reset* is enabled, variables preserved from the invoking user's environment by the *env_keep* list take precedence over those in the PAM environment. When *v_reset* is disabled, variables present in the invoking user's environment take precedence over those in the PAM environment unless they match a pattern in the *env_delete* list.

Note that the dynamic linker on most operating systems will remove variables that can control dynamic linking from the environment of set-user-ID executables, including **sudo**. Depending on the operating system this may include _RLD*, DYLD_*, LD_*, LDR_*, LIBPATH, SHLIB_PATH, and others. These type of variables are removed from the environment before **sudo** even begins execution and, as such, it is not possible for **sudo** to preserve them.

As a special case, if the **-i** option (initial login) is specified, **sudoers** will initialize the environment regardless of the value of *env_reset*. The DISPLAY, PATH and TERM variables remain unchanged; HOME, MAIL, SHELL, USER, and LOGNAME are set based on the target user. On AIX (and Linux systems without PAM), the contents of /etc/environment are also included. All other environment variables are removed unless permitted by *env_keep* or *env_check*, described above.

Finally, the *restricted_env_file* and *env_file* files are applied, if present. The variables in *restricted_env_file* are applied first and are subject to the same restrictions as the invoking user's environment, as detailed above. The variables in *env_file* are applied last and are not subject to these restrictions. In both cases, variables present in the files will only be set to their specified values if they would not conflict with an existing environment variable.

SUDOERS FILE FORMAT

The *sudoers* file is composed of two types of entries: aliases (basically variables) and user specifications (which specify who may run what).

When multiple entries match for a user, they are applied in order. Where there are multiple matches, the last match is used (which is not necessarily the most specific match).

The *sudoers* file grammar will be described below in Extended Backus-Naur Form (EBNF). Don't despair if you are unfamiliar with EBNF; it is fairly simple, and the definitions below are annotated.

Resource limits

By default, **sudoers** uses the operating system's native method of setting resource limits for the target user. On Linux systems, resource limits are usually set by the *pam_limits.so* PAM module. On some BSD systems, the */etc/login.conf* file specifies resource limits for the user. On AIX systems, resource limits are configured in the */etc/security/limits* file. If there is no system mechanism to set per-user resource limits, the command will run with the same limits as the invoking user. The one exception to this is the core dump file size, which is set by **sudoers** to 0 by default. Disabling core dumps by default makes it possible to avoid potential security problems where the core file is treated as trusted input.

Resource limits may also be set in the *sudoers* file itself, in which case they override those set by the system. See the *rlimit_as*, *rlimit_core*, *rlimit_cpu*, *rlimit_data*, *rlimit_fsize*, *rlimit_locks*, *rlimit_memlock*, *rlimit_nofile*, *rlimit_nproc*, *rlimit_rss*, *rlimit_stack* options described below. Resource limits in **sudoers** may be specified in one of the following formats:

“value”

Both the soft and hard resource limits are set to the same value. The special value “infinity” can be used to indicate that the value is unlimited.

“soft,hard”

Two comma-separated values. The soft limit is set to the first value and the hard limit is set to the second. Both values must either be enclosed in a set of double quotes, or the comma must be escaped with a backslash (‘\’). The special value “infinity” may be used in place of either value.

“default”

The default resource limit for the user will be used. This may be a user-specific value (see above) or the value of the resource limit when **sudo** was invoked for systems that don't support per-user limits.

“user” The invoking user's resource limits will be preserved when running the command.

For example, to restore the historic core dump file size behavior, a line like the following may be used.

```
Defaults rlimit_core=default
```

Resource limits in **sudoers** are only supported by version 1.8.7 or higher.

Quick guide to EBNF

EBNF is a concise and exact way of describing the grammar of a language. Each EBNF definition is made up of *production rules*. E.g.,

```
symbol ::= definition | alternate1 | alternate2 . . .
```

Each *production rule* references others and thus makes up a grammar for the language. EBNF also contains the following operators, which many readers will recognize from regular expressions. Do not, however, confuse them with “wildcard” characters, which have different meanings.

- ? Means that the preceding symbol (or group of symbols) is optional. That is, it may appear once or not at all.
- * Means that the preceding symbol (or group of symbols) may appear zero or more times.
- + Means that the preceding symbol (or group of symbols) may appear one or more times.

Parentheses may be used to group symbols together. For clarity, we will use single quotes (‘ ’) to designate what is a verbatim character string (as opposed to a symbol name).

Aliases

There are four kinds of aliases: User_Alias, Runas_Alias, Host_Alias and Cmnd_Alias. Beginning with **sudo** 1.9.0, Cmd_Alias may be used in place of Cmnd_Alias if desired.

```
Alias ::= 'User_Alias' User_Alias_Spec (':' User_Alias_Spec)* |  
       'Runas_Alias' Runas_Alias_Spec (':' Runas_Alias_Spec)* |  
       'Host_Alias' Host_Alias_Spec (':' Host_Alias_Spec)* |  
       'Cmnd_Alias' Cmnd_Alias_Spec (':' Cmnd_Alias_Spec)* |  
       'Cmd_Alias' Cmnd_Alias_Spec (':' Cmnd_Alias_Spec)*
```

```
User_Alias ::= NAME
```

```
User_Alias_Spec ::= User_Alias '=' User_List
```

```
Runas_Alias ::= NAME
```

```
Runas_Alias_Spec ::= Runas_Alias '=' Runas_List
```

```
Host_Alias ::= NAME
```

```
Host_Alias_Spec ::= Host_Alias '=' Host_List
```

```
Cmnd_Alias ::= NAME
```

```
Cmnd_Alias_Spec ::= Cmnd_Alias '=' Cmnd_List
```

```
NAME ::= [A-Z]([A-Z][0-9]_)*
```

Each *alias* definition is of the form

```
Alias_Type NAME = item1, item2, ...
```

where *Alias_Type* is one of User_Alias, Runas_Alias, Host_Alias, or Cmnd_Alias. ANAME is a string of uppercase letters, numbers, and underscore characters (‘ _ ’). ANAME **must** start with an uppercase letter. It is possible to put several alias definitions of the same type on a single line, joined by a colon (‘ : ’). E.g.,

```
Alias_Type NAME = item1, item2, item3 : NAME = item4, item5
```

It is a syntax error to redefine an existing *alias*. It is possible to use the same name for *aliases* of different types, but this is not recommended.

The definitions of what constitutes a valid *alias* member follow.

```
User_List ::= User |
             User ',' User_List

User ::= '!'* user_name |
           '!'* #user-ID |
           '!'* %group |
           '!'* %#group-ID |
           '!'* +netgroup |
           '!'* %:nonunix_group |
           '!'* %:#nonunix_gid |
           '!'* User_Alias
```

A *User_List* is made up of one or more user names, user-IDs (prefixed with '#'), system group names and IDs (prefixed with '%' and '%#' respectively), netgroups (prefixed with '+'), non-Unix group names and IDs (prefixed with '%:' and%:# respecti vely), and *User_Aliases*. Each list item may be prefixed with zero or more '!' operators. An odd number of '!' operators negate the value of the item; an even number just cancel each other out. User netgroups are matched using the user and domain members only; the host member is not used when matching.

A *user_name*, *user-ID*, *group*, *group-ID*, *netgroup*, *nonunix_group* or *nonunix_gid* may be enclosed in double quotes to avoid the need for escaping special characters. Alternately, special characters may be specified in escaped hex mode, e.g., \x20 for space. When using double quotes, any prefix characters must be included inside the quotes.

The actual *nonunix_group* and *nonunix_gid* syntax depends on the underlying group provider plugin. For instance, the QAS AD plugin supports the following formats:

- Group in the same domain: "%:Group Name"
- Group in any domain: "%:Group Name@FULLY.QUALIFIED.DOMAIN"
- Group SID: "%:S-1-2-34-5678901234-5678901234-5678901234-567"

See **GROUP PROVIDER PLUGINS** for more information.

Note that quotes around group names are optional. Unquoted strings must use a backslash ('\') to escape spaces and special characters. See **Other special characters and reserved words** for a list of characters that need to be escaped.

```
Runas_List ::= Runas_Member |
               Runas_Member ',' Runas_List

Runas_Member ::= '!'* user_name |
                  '!'* #user-ID |
                  '!'* %group |
                  '!'* %#group-ID |
                  '!'* %:nonunix_group |
                  '!'* %:#nonunix_gid |
                  '!'* +netgroup |
                  '!'* Runas_Alias
```

A *Runas_List* is similar to a *User_List* except that instead of *User_Aliases* it can contain *Runas_Aliases*. Note that user names and groups are matched as strings. In other words, two users (groups) with the same user (group) ID are considered to be distinct. If you wish to match all user names with the same user-ID (e.g., root and toor), you can use a user-ID instead of a name (#0 in the example

given). Note that the user-ID or group-ID specified in a Runas_Member need not be listed in the password or group database.

```
Host_List ::= Host |
              Host ',' Host_List

Host ::= '!'* host name |
          '!'* ip_addr |
          '!'* network(/netmask)? |
          '!'* +netgroup |
          '!'* Host_Alias
```

A Host_List is made up of one or more host names, IP addresses, network numbers, netgroups (prefixed with '+'), and other aliases. Again, the value of an item may be negated with the '!' operator. Host netgroups are matched using the host (both qualified and unqualified) and domain members only; the user member is not used when matching. If you specify a network number without a netmask, **sudo** will query each of the local host's network interfaces and, if the network number corresponds to one of the hosts's network interfaces, will use the netmask of that interface. The netmask may be specified either in standard IP address notation (e.g., 255.255.255.0 or ffff:ffff:ffff:ffff::), or CIDR notation (number of bits, e.g., 24 or 64). A host name may include shell-style wildcards (see the **Wildcards** section below), but unless the host name command on your machine returns the fully qualified host name, you'll need to use the *fqdn* flag for wildcards to be useful. Note that **sudo** only inspects actual network interfaces; this means that IP address 127.0.0.1 (localhost) will never match. Also, the host name "localhost" will only match if that is the actual host name, which is usually only the case for non-networked systems.

```
digest ::= [A-Fa-f0-9]+ |
          [A-Za-z0-9\+/_]+

Digest_Spec ::= "sha224" ':: digest |
                  "sha256" ':: digest |
                  "sha384" ':: digest |
                  "sha512" ':: digest

Digest_List ::= Digest_Spec |
                  Digest_Spec ',' Digest_List

Cmnd_List ::= Cmnd |
                  Cmnd ',' Cmnd_List

command name ::= file name |
                  file name args |
                  file name '"'

Edit_Spec ::= "sudoedit" file name+

Cmnd ::= Digest_List? '!'* command name |
          '!'* directory |
          '!'* Edit_Spec |
          '!'* Cmnd_Alias
```

A Cmnd_List is a list of one or more command names, directories, and other aliases. A command name is a fully qualified file name which may include shell-style wildcards (see the **Wildcards** section below). A simple file name allows the user to run the command with any arguments they wish. However, you may also specify command line arguments (including wildcards). Alternately, you can specify "" to indicate that the

command may only be run **without** command line arguments. A directory is a fully qualified path name ending in a ‘/’. When you specify a directory in a Cmnd_List, the user will be able to run any file within that directory (but not in any sub-directories therein).

If a Cmnd has associated command line arguments, then the arguments in the Cmnd must match exactly those given by the user on the command line (or match the wildcards if there are any). Note that the following characters must be escaped with a ‘\’ if they are used in command arguments: ‘,’ ‘:’ ‘=’ ‘\’. The built-in command “**sudoedit**” is used to permit a user to run **sudo** with the **-e** option (or as **sudoedit**). It may take command line arguments just as a normal command does. Note that “**sudoedit**” is a command built into **sudo** itself and must be specified in the *sudoers* file **without** a leading path. If a leading path is present, for example /usr/bin/sudoedit, the path name will be silently converted to “**sudoedit**”. A fully-qualified path for **sudoedit** is treated as an error by **visudo**.

A command name may be preceded by a Digest_List, a comma-separated list of one or more Digest_Spec entries. If a Digest_List is present, the command will only match successfully if it can be verified using one of the SHA-2 digests in the list. Starting with version 1.9.0, the **ALL** reserved word can be used in conjunction with a Digest_List. The following digest formats are supported: sha224, sha256, sha384, and sha512. The string may be specified in either hex or base64 format (base64 is more compact). There are several utilities capable of generating SHA-2 digests in hex format such as openssl, shasum, sha224sum, sha256sum, sha384sum, sha512sum.

For example, using openssl:

```
$ openssl dgst -sha224 /bin/ls
SHA224(/bin/ls)= 118187da8364d490b4a7debbf483004e8f3e053ec954309de2c41a25
```

It is also possible to use openssl to generate base64 output:

```
$ openssl dgst -binary -sha224 /bin/ls | openssl base64
EYGH2oNk1JC0p9679IMATo8+BT7JVDCd4sQaJQ==
```

Warning, if the user has write access to the command itself (directly or via a **sudo** command), it may be possible for the user to replace the command after the digest check has been performed but before the command is executed. A similar race condition exists on systems that lack the **fexecve()** system call when the directory in which the command is located is writable by the user. See the description of the *fde_xec* setting for more information on how **sudo** executes commands that have an associated digest.

Command digests are only supported by version 1.8.7 or higher.

Defaults

Certain configuration options may be changed from their default values at run-time via one or more Default_Entry lines. These may affect all users on any host, all users on a specific host, a specific user, a specific command, or commands being run as a specific user. Note that per-command entries may not include command line arguments. If you need to specify arguments, define a Cmnd_Alias and reference that instead.

```
Default_Type ::= 'Defaults' |
                 'Defaults' '@' Host_List |
                 'Defaults' ':' User_List |
                 'Defaults' '!' Cmnd_List |
                 'Defaults' '>' Runas_List

Default_Entry ::= Default_Type Parameter_List

Parameter_List ::= Parameter |
                  Parameter ',' Parameter_List
```

```
Parameter ::= Parameter '=' Value |
             Parameter '+=' Value |
             Parameter '-=' Value |
             '!' * Parameter
```

Parameters may be **flags**, **integer** values, **strings**, or **lists**. Flags are implicitly boolean and can be turned off via the ‘!’ operator. Some integer, string and list parameters may also be used in a boolean context to disable them. Values may be enclosed in double quotes (“”) when they contain multiple words. Special characters may be escaped with a backslash (‘\’).

To include a literal backslash character in a command line argument you must escape the backslash twice. For example, to match ‘\n’ as part of a command line argument, you must use ‘\\n’ in the *sudoers* file. This is due to there being two levels of escaping, one in the *sudoers* parser itself and another when command line arguments are matched by the *fnmatch(3)* function.

Lists have two additional assignment operators, += and -=. These operators are used to add to and delete from a list respectively. It is not an error to use the-= operator to remove an element that does not exist in a list.

Defaults entries are parsed in the following order: generic, host, user, and runas Defaults first, then command defaults. If there are multiple Defaults settings of the same type, the last matching setting is used. The following Defaults settings are parsed before all others since they may affect subsequent entries: *fqdn*, *group_plugin*, *runas_default*, *sudoers_locale*.

See **SUDOERS OPTIONS** for a list of supported Defaults parameters.

User specification

```
User_Spec ::= User_List Host_List '=' Cmnd_Spec_List \
             (':' Host_List '=' Cmnd_Spec_List)*

Cmnd_Spec_List ::= Cmnd_Spec |
                   Cmnd_Spec ',' Cmnd_Spec_List

Cmnd_Spec ::= Runas_Spec? Option_Spec* Tag_Spec* Cmnd

Runas_Spec ::= '(' Runas_List? (':' Runas_List)? ')'

Option_Spec ::= (SELinux_Spec | Date_Spec | Timeout_Spec | Chdir_Spec | Chroot_Spec)

SELinux_Spec ::= ('ROLE=role' | 'TYPE=type')

Date_Spec ::= ('NOTBEFORE=timestamp' | 'NOTAFTER=timestamp')

Timeout_Spec ::= 'TIMEOUT=timeout'

Chdir_Spec ::= 'CWD=directory'

Chroot_Spec ::= 'CHROOT=directory'

Tag_Spec ::= ('EXEC:' | 'NOEXEC:' | 'FOLLOW:' | 'NOFOLLOW' |
              'LOG_INPUT:' | 'NOLOG_INPUT:' | 'LOG_OUTPUT:' |
              'NOLOG_OUTPUT:' | 'MAIL:' | 'NOMAIL:' | 'INTERCEPT:' |
              'NOINTERCEPT:' | 'PASSWD:' | 'NOPASSWD:' | 'SETENV:' |
              'NOSETENV:')
```

A **user specification** determines which commands a user may run (and as what user) on specified hosts. By default, commands are run as **root**, but this can be changed on a per-command basis.

The basic structure of a user specification is “who where = (as_whom) what”. Let’s break that down into its constituent parts:

Runas_Spec

A Runas_Spec determines the user and/or the group that a command may be run as. A fully-specified Runas_Spec consists of two Runas_Lists (as defined above) separated by a colon (‘:’) and enclosed in a set of parentheses. The first Runas_List indicates which users the command may be run as via the **-u** option. The second defines a list of groups that may be specified via the **-g** option (in addition to any of the target user’s groups). If both Runas_Lists are specified, the command may be run with any combination of users and groups listed in their respective Runas_Lists. If only the first is specified, the command may be run as any user in the list and, optionally, with any group the target user belongs to. If the first Runas_List is empty but the second is specified, the command may be run as the invoking user with the group set to any listed in the Runas_List. If both Runas_Lists are empty, the command may only be run as the invoking user and the group, if specified, must be one that the invoking user is a member of. If no Runas_Spec is specified, the command may only be run as **root** and the group, if specified, must be one that **root** is a member of.

A Runas_Spec sets the default for the commands that follow it. What this means is that for the entry:

```
dgb      boulder = (operator) /bin/ls, /bin/kill, /usr/bin/lprm
```

The user **dgb** may run `/bin/ls`, `/bin/kill`, and `/usr/bin/lprm` on the host `boulder`—but only as **operator**. E.g.,

```
$ sudo -u operator /bin/ls
```

It is also possible to override a Runas_Spec later on in an entry. If we modify the entry like so:

```
dgb      boulder = (operator) /bin/ls, (root) /bin/kill, /usr/bin/lprm
```

Then user **dgb** is now allowed to run `/bin/ls` as **operator**, but `/bin/kill` and `/usr/bin/lprm` as **root**.

We can extend this to allow **dgb** to run `/bin/ls` with either the user or group set to **operator**:

```
dgb      boulder = (operator : operator) /bin/ls, (root) /bin/kill,\n      /usr/bin/lprm
```

Note that while the group portion of the Runas_Spec permits the user to run as command with that group, it does not force the user to do so. If no group is specified on the command line, the command will run with the group listed in the target user’s password database entry. The following would all be permitted by the sudoers entry above:

```
$ sudo -u operator /bin/ls\n$ sudo -u operator -g operator /bin/ls\n$ sudo -g operator /bin/ls
```

In the following example, user **tcm** may run commands that access a modem device file with the dialer group.

```
tcm      boulder = (:dialer) /usr/bin/tip, /usr/bin/cu,\n      /usr/local/bin/minicom
```

Note that in this example only the group will be set, the command still runs as user **tcm**. E.g.

```
$ sudo -g dialer /usr/bin/cu
```

Multiple users and groups may be present in a Runas_Spec, in which case the user may select any combination of users and groups via the **-u** and **-g** options. In this example:

```
alan    ALL = (root, bin : operator, system) ALL
```

user **alan** may run any command as either user root or bin, optionally setting the group to operator or system.

Option_Spec

A Cmnd may have zero or more options associated with it. Options may consist of SELinux roles and/or types, start and/or end dates and command timeouts. Once an option is set for a Cmnd, subsequent Cmrnds in the Cmnd_Spec_List, inherit that option unless it is overridden by another option. Note that the option names are reserved words in *sudoers*. This means that none of the valid option names (see below) can be used when declaring an alias.

SELinux_Spec

On systems with SELinux support, *sudoers* file entries may optionally have an SELinux role and/or type associated with a command. This can be used to implement a form of role-based access control (RBAC). If a role or type is specified with the command it will override any default values specified in *sudoers*. A role or type specified on the command line, however, will supersede the values in *sudoers*.

Date_Spec

sudoers rules can be specified with a start and end date via the NOTBEFORE and NOTAFTER settings. The time stamp must be specified in *Generalized Time* as defined by RFC 4517. The format is effectively `yyyymmddHHMMSSZ` where the minutes and seconds are optional. The ‘z’ suffix indicates that the time stamp is in Coordinated Universal Time (UTC). It is also possible to specify a timezone offset from UTC in hours and minutes instead of a ‘z’. For example, `-0500` would correspond to Eastern Standard time in the US. As an extension, if no ‘z’ or timezone offset is specified, local time will be used.

The following are all valid time stamps:

```
20170214083000Z
2017021408Z
20160315220000-0500
20151201235900
```

Timeout_Spec

A command may have a timeout associated with it. If the timeout expires before the command has exited, the command will be terminated. The timeout may be specified in combinations of days, hours, minutes, and seconds with a single-letter case-insensitive suffix that indicates the unit of time. For example, a timeout of 7 days, 8 hours, 30 minutes, and 10 seconds would be written as `7d8h30m10s`. If a number is specified without a unit, seconds are assumed. Any of the days, minutes, hours, or seconds may be omitted. The order must be from largest to smallest unit and a unit may not be specified more than once.

The following are all *valid* timeout values: `7d8h30m10s`, `14d`, `8h30m`, `600s`, `3600`. The following are *invalid* timeout values: `12m2w1d`, `30s10m4h`, `1d2d3h`.

This setting is only supported by version 1.8.20 or higher.

Chdir_Spec

The working directory that the command will be run in can be specified using the CWD setting. The *directory* must be a fully-qualified path name beginning with a ‘/’ or ‘~’ character, or the special value “*”. A value of “*” indicates that the user may specify the working directory by running **sudo** with the **-D** option. By default, commands are run from the invoking user’s current working directory, unless the **-i** op-

tion is given. Path names of the form `~user/path/name` are interpreted as being relative to the named user's home directory. If the user name is omitted, the path will be relative to the runas user's home directory.

This setting is only supported by version 1.9.3 or higher.

Chroot_Spec

The root directory that the command will be run in can be specified using the CHROOT setting. The *directory* must be a fully-qualified path name beginning with a '/' or '~' character, or the special value '*'. A value of '*' indicates that the user may specify the root directory by running **sudo** with the **-R** option. This setting can be used to run the command in a chroot(2) "sandbox" similar to the chroot(8) utility. Path names of the form `~user/path/name` are interpreted as being relative to the named user's home directory. If the user name is omitted, the path will be relative to the runas user's home directory.

This setting is only supported by version 1.9.3 or higher.

Tag_Spec

A command may have zero or more tags associated with it. The following tag values are supported: EXEC, NOEXEC, FOLLOW, NOFOLLOW, LOG_INPUT, NOLOG_INPUT, LOG_OUTPUT, NOLOG_OUTPUT, MAIL, NOMAIL, INTERCEPT, NOINTERCEPT, PASSWD, NOPASSWD, SETENV, and NOSETENV. Once a tag is set on a Cmnd, subsequent Cmnds in the Cmnd_Spec_List, inherit the tag unless it is overridden by the opposite tag (in other words, PASSWD overrides NOPASSWD and NOEXEC overrides EXEC).

EXEC and *NOEXEC*

If **sudo** has been compiled with *noexec* support and the underlying operating system supports it, the NOEXEC tag can be used to prevent a dynamically-linked executable from running further commands itself.

In the following example, user **aaron** may run `/usr/bin/more` and `/usr/bin/vi` but shell escapes will be disabled.

```
aaron    shanty = NOEXEC: /usr/bin/more, /usr/bin/vi
```

See the **Preventing shell escapes** section below for more details on how NOEXEC works and whether or not it will work on your system.

FOLLOW and *NOFOLLOW* Starting with version 1.8.15, **sudoedit** will not open a file that is a symbolic link unless the *sudoedit_follow* flag is enabled. The *FOLLOW* and *NOFOLLOW* tags override the value of *sudoedit_follow* and can be used to permit (or deny) the editing of symbolic links on a per-command basis. These tags are only effective for the **sudoedit** command and are ignored for all other commands.

LOG_INPUT and *NOLOG_INPUT*

These tags override the value of the *log_input* flag on a per-command basis. For more information, see the description of *log_input* in the **SUDOERS OPTIONS** section below.

LOG_OUTPUT and *NOLOG_OUTPUT*

These tags override the value of the *log_output* flag on a per-command basis. For more information, see the description of *log_output* in the **SUDOERS OPTIONS** section below.

MAIL and *NOMAIL*

These tags provide fine-grained control over whether mail will be sent when a user runs a command by overriding the value of the *mail_all_cmnds* flag on a per-command basis. They have no effect when **sudo** is run with the **-1** or **-v** options. A *NOMAIL* tag will also override the *mail_always* and *mail_no_perms* options. For more information, see the descriptions of *mail_all_cmnds*, *mail_always*, and *mail_no_perms*.

in the **SUDOERS OPTIONS** section below.

PASSWD and *NOPASSWD*

By default, **sudo** requires that a user authenticate before running a command. This behavior can be modified via the **NOPASSWD** tag. Like a **Runas_Spec**, the **NOPASSWD** tag sets a default for the commands that follow it in the **Cmnd_Spec_List**. Conversely, the **PASSWD** tag can be used to reverse things. For example:

```
ray      rushmore = NOPASSWD: /bin/kill, /bin/ls, /usr/bin/lprm
```

would allow the user **ray** to run **/bin/kill**, **/bin/ls**, and **/usr/bin/lprm** as **root** on the machine “rushmore” without authenticating himself. If we only want **ray** to be able to run **/bin/kill** without a password the entry would be:

```
ray      rushmore = NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lprm
```

Note, however, that the **PASSWD** tag has no effect on users who are in the group specified by the **exempt_group** setting.

By default, if the **NOPASSWD** tag is applied to any of a user’s entries for the current host, the user will be able to run “**sudo -l**” without a password. Additionally, a user may only run “**sudo -v**” without a password if all of the user’s entries for the current host have the **NOPASSWD** tag. This behavior may be overridden via the **verifypw** and **listpw** options.

SETENV and *NOSETENV*

These tags override the value of the *setenv* flag on a per-command basis. Note that if **SETENV** has been set for a command, the user may disable the *env_reset* flag from the command line via the **-E** option. Additionally, environment variables set on the command line are not subject to the restrictions imposed by *env_check*, *env_delete*, or *env_keep*. As such, only trusted users should be allowed to set variables in this manner. If the command matched **isALL**, the **SETENV** tag is implied for that command; this default may be overridden by use of the **NOSETENV** tag.

INTERCEPT and *NOINTERCEPT*

If **sudo** has been compiled with *intercept* support and the underlying operating system supports it, the **INTERCEPT** tag can be used to cause programs spawned by a command to be validated against *sudoers* and logged just like they would be if run through **sudo** directly. This is useful in conjunction with commands that allow shell escapes such as editors, shells, and paginators.

In the following example, user **chuck** may run any command on the machine “research” in intercept mode.

```
chuck  research = INTERCEPT: ALL
```

See the **Preventing shell escapes** section below for more details on how **INTERCEPT** works and whether or not it will work on your system.

Wildcards

sudo allows shell-style *wildcards* (aka meta or glob characters) to be used in host names, path names, and command line arguments in the *sudoers* file. Wildcard matching is done via the **glob(3)** and **fnmatch(3)** functions as specified by IEEE Std 1003.1 (“POSIX.1”).

- * Matches any set of zero or more characters (including white space).
- ? Matches any single character (including white space).

- [. . .] Matches any character in the specified range.
- [! . . .] Matches any character *not* in the specified range.
- \x For any character ‘x’, evaluates to ‘x’. This is used to escape special characters such as ‘*’, ‘?’, ‘[’, and ‘]’.

Note that these are not regular expressions. Unlike a regular expression there is no way to match one or more characters within a range.

Character classes may be used if your system’s `glob(3)` and `fnmatch(3)` functions support them. However, because the ‘:’ character has special meaning in *sudoers*, it must be escaped. For example:

```
/bin/ls [[\:alpha\:]]*
```

Would match any file name beginning with a letter.

Note that a forward slash (‘/’) will *not* be matched by wildcards used in the file name portion of the command. This is to make a path like:

```
/usr/bin/*
```

match /usr/bin/who but not /usr/bin/X11/xterm.

When matching the command line arguments, however, a slash *does* get matched by wildcards since command line arguments may contain arbitrary strings and not just path names.

Wildcards in command line arguments should be used with care.

Command line arguments are matched as a single, concatenated string. This means a wildcard character such as ‘?’ or ‘*’ will match across word boundaries, which may be unexpected. For example, while a sudoers entry like:

```
%operator ALL = /bin/cat /var/log/messages*
```

will allow command like:

```
$ sudo cat /var/log/messages.1
```

It will also allow:

```
$ sudo cat /var/log/messages /etc/shadow
```

which is probably not what was intended. In most cases it is better to do command line processing outside of the *sudoers* file in a scripting language.

Exceptions to wildcard rules

The following exceptions apply to the above rules:

- "" If the empty string “” is the only command line argument in the *sudoers* file entry it means that command is not allowed to be run with *any* arguments.
- sudoedit Command line arguments to the *sudoedit* built-in command should always be path names, so a forward slash (‘/’) will not be matched by a wildcard.

Including other files from within sudoers

It is possible to include other *sudoers* files from within the *sudoers* file currently being parsed using the `@include` and `@includedir` directives. For compatibility with sudo versions prior to 1.9.1, `#include` and `#includedir` are also accepted.

An include file can be used, for example, to keep a site-wide *sudoers* file in addition to a local, per-machine file. For the sake of this example the site-wide *sudoers* file will be `/etc/sudoers` and the per-machine one will be `/etc/sudoers.local`. To include `/etc/sudoers.local` from within

/etc/sudoers one would use the following line in /etc/sudoers:

```
@include /etc/sudoers.local
```

When **sudo** reaches this line it will suspend processing of the current file (/etc/sudoers) and switch to /etc/sudoers.local. Upon reaching the end of /etc/sudoers.local, the rest of /etc/sudoers will be processed. Files that are included may themselves include other files. A hard limit of 128 nested include files is enforced to prevent include file loops.

Starting with version 1.9.1, the path to the include file may contain white space if it is escaped with a backslash ('\'). Alternately, the entire path may be enclosed in double quotes (""), in which case no escaping is necessary. To include a literal backslash in the path, '\\\' should be used.

If the path to the include file is not fully-qualified (does not begin with a '/'), it must be located in the same directory as the sudoers file it was included from. For example, if /etc/sudoers contains the line:

```
@include sudoers.local
```

the file that will be included is /etc/sudoers.local.

The file name may also include the %h escape, signifying the short form of the host name. In other words, if the machine's host name is "xerxes", then

```
@include /etc/sudoers.%h
```

will cause **sudo** to include the file /etc/sudoers.xerxes.

The @includedir directive can be used to create a sudoers.d directory that the system package manager can drop sudoers file rules into as part of package installation. For example, given:

```
@includedir /etc/sudoers.d
```

sudo will suspend processing of the current file and read each file in /etc/sudoers.d, skipping file names that end in '~' or contain a '.' character to avoid causing problems with package manager or editor temporary/backup files. Files are parsed in sorted lexical order. That is, /etc/sudoers.d/01_first will be parsed before /etc/sudoers.d/10_second. Be aware that because the sorting is lexical, not numeric, /etc/sudoers.d/1_whoops would be loaded after /etc/sudoers.d/10_second. Using a consistent number of leading zeroes in the file names can be used to avoid such problems. After parsing the files in the directory, control returns to the file that contained the @includedir directive.

Note that unlike files included via @include, visudo will not edit the files in a @includedir directory unless one of them contains a syntax error. It is still possible to runvisudo with the -f flag to edit the files directly, but this will not catch the redefinition of an alias that is also present in a different file.

Other special characters and reserved words

The pound sign ('#') is used to indicate a comment (unless it is part of a #include directive or unless it occurs in the context of a user name and is followed by one or more digits, in which case it is treated as a user-ID). Both the comment character and any text after it, up to the end of the line, are ignored.

The reserved word **ALL** is a built-in alias that always causes a match to succeed. It can be used wherever one might otherwise use a Cmnd_Alias, User_Alias, Runas_Alias, or Host_Alias. Attempting to define an alias named **ALL** will result in a syntax error. Please note that using**ALL** can be dangerous since in a command context, it allows the user to run *any* command on the system.

The following option names permitted in an Option_Spec are also considered reserved words: CHROOT, ROLE, TYPE, TIMEOUT, CWD, NOTBEFORE and NOTAFTER. Attempting to define *alias* with the same name as one of the options will result in a syntax error.

An exclamation point ('!') can be used as a logical *not* operator in a list or *alias* as well as in front of a *Cmnd*. This allows one to exclude certain values. For the '!' operator to be effective, there must be something for it to exclude. For example, to match all users except for root one would use:

```
ALL, !root
```

If the **ALL**, is omitted, as in:

```
!root
```

it would explicitly deny root but not match any other users. This is different from a true “negation” operator.

Note, however, that using a '!' in conjunction with the built-in **ALL** alias to allow a user to run “all but a few” commands rarely works as intended (see **SECURITY NOTES** below).

Long lines can be continued with a backslash ('\') as the last character on the line.

White space between elements in a list as well as special syntactic characters in a *User Specification* ('=', ':', '(', ')') is optional.

The following characters must be escaped with a backslash ('\') when used as part of a word (e.g., a user name or host name): '!', '=', ':', ',', '(', ')', '\'.

SUDOERS OPTIONS

sudo's behavior can be modified by *Default_Entry* lines, as explained earlier. A list of all supported Defaults parameters, grouped by type, are listed below.

Boolean Flags:

always_query_group_plugin

If a *group_plugin* is configured, use it to resolve groups of the form %group as long as there is not also a system group of the same name. Normally, only groups of the form %:group are passed to the *group_plugin*. This flag is *off* by default.

always_set_home

If enabled, **sudo** will set the HOME environment variable to the home directory of the target user (which is the root user unless the **-u** option is used). This flag is largely obsolete and has no effect unless the *env_reset* flag has been disabled or HOME is present in the *env_keep* list, both of which are strongly discouraged. This flag is *off* by default.

authenticate

If set, users must authenticate themselves via a password (or other means of authentication) before they may run commands. This default may be overridden via the PASSWD and NOPASSWD tags. This flag is *on* by default.

case_insensitive_group

If enabled, group names in *sudoers* will be matched in a case insensitive manner. This may be necessary when users are stored in LDAP or AD. This flag is *on* by default.

case_insensitive_user

If enabled, user names in *sudoers* will be matched in a case insensitive manner. This may be necessary when groups are stored in LDAP or AD. This flag is *on* by default.

closefrom_override

If set, the user may use the **-C** option which overrides the default starting point at which **sudo** begins closing open file descriptors. This flag is *off* by default.

compress_io

If set, and **sudo** is configured to log a command's input or output, the I/O logs will be compressed using **zlib**. This flag is *on* by default when **sudo** is compiled with **zlib** support.

exec_background

By default, **sudo** runs a command as the foreground process as long as **sudo** itself is running in the foreground. When the *exec_background* flag is enabled and the command is being run in a pseudo-terminal (due to I/O logging or the *use_pty* flag), the command will be run as a background process. Attempts to read from the controlling terminal (or to change terminal settings) will result in the command being suspended with the SIGTTIN signal (or SIGTTOU in the case of terminal settings). If this happens when **sudo** is a foreground process, the command will be granted the controlling terminal and resumed in the foreground with no user intervention required. The advantage of initially running the command in the background is that **sudo** need not read from the terminal unless the command explicitly requests it. Otherwise, any terminal input must be passed to the command, whether it has required it or not (the kernel buffers terminals so it is not possible to tell whether the command really wants the input). This is different from historic *sudo* behavior or when the command is not being run in a pseudo-terminal.

For this to work seamlessly, the operating system must support the automatic restarting of system calls. Unfortunately, not all operating systems do this by default, and even those that do may have bugs. For example, macOS fails to restart the **tcgetattr()** and **tcsetattr()** system calls (this is a bug in macOS). Furthermore, because this behavior depends on the command stopping with the SIGTTIN or SIGTTOU signals, programs that catch these signals and suspend themselves with a different signal (usually SIGTOP) will not be automatically foregrounded. Some versions of the linux **su(1)** command behave this way. This flag is off by default.

This setting is only supported by version 1.8.7 or higher. It has no effect unless I/O logging is enabled or the *use_pty* flag is enabled.

env_editor

If set, **visudo** will use the value of the SUDO_EDITOR, VISUAL or EDITOR environment variables before falling back on the default editor list. Note that **visudo** is typically run as root so this flag may allow a user with **visudo** privileges to run arbitrary commands as root without logging. An alternative is to place a colon-separated list of “safe” editors int the *editor* setting. **visudo** will then only use SUDO_EDITOR, VISUAL or EDITOR if they match a value specified in *editor*. If the *v_reset* flag is enabled, the SUDO_EDITOR, VISUAL and/or EDITOR environment variables must be present in the *env_keep* list for the *env_editor* flag to function when **visudo** is invoked via **sudo**. This flag is on by default.

env_reset

If set, **sudo** will run the command in a minimal environment containing the TERM, PATH, HOME, MAIL, SHELL, LOGNAME, USER and SUDO_* variables. Any variables in the caller’s environment or in the file specified by the *restricted_env_file* setting that match the *env_keep* and *env_check* lists are then added, followed by any variables present in the file specified by the *env_file* setting (if any). The contents of the *env_keep* and *env_check* lists, as modified by global Defaults parameters in *sudoers*, are displayed when **sudo** is run by root with the **-v** option. If the *secure_path* setting is enabled, its value will be used for the PATH environment variable. This flag is off by default.

fast_glob

Normally, **sudo** uses the **glob(3)** function to do shell-style globbing when matching path names. However, since it accesses the file system, **glob(3)** can take a long time to complete for some patterns, especially when the pattern references a network file system that is mounted on demand (auto mounted). The *fast_glob* flag causes **sudo** to use the **fnmatch(3)** function, which does not access the file system to do its matching. The disadvantage of *fast_glob* is that it is unable to match relative path names such as *./ls* or *../bin/ls*. This has security implications when path names that include

globbing characters are used with the negation operator, ‘!’, as such rules can be trivially bypassed. As such, this flag should not be used when the *sudoers* file contains rules that contain negated path names which include globbing characters. This flag is *off* by default.

fqdn

Set this flag if you want to put fully qualified host names in the *sudoers* file when the local host name (as returned by the *hostname* command) does not contain the domain name. In other words, instead of myhost you would use myhost.mydomain.edu. You may still use the short form if you wish (and even mix the two). This flag is only effective when the “canonical” host name, as returned by the **getaddrinfo()** or **gethostbyname()** function, is a fully-qualified domain name. This is usually the case when the system is configured to use DNS for host name resolution.

If the system is configured to use the */etc/hosts* file in preference to DNS, the “canonical” host name may not be fully-qualified. The order that sources are queried for host name resolution is usually specified in the */etc/nsswitch.conf*, */etc/netsvc.conf*, */etc/host.conf*, or, in some cases, */etc/resolv.conf* file. In the */etc/hosts* file, the first host name of the entry is considered to be the “canonical” name; subsequent names are aliases that are not used by **sudoers**. For example, the following hosts file line for the machine “xyzzy” has the fully-qualified domain name as the “canonical” host name, and the short version as an alias.

```
192.168.1.1 xyzzy.sudo.ws xyzzy
```

If the machine’s hosts file entry is not formatted properly, the *fqdn* flag will not be effective if it is queried before DNS.

Beware that when using DNS for host name resolution, turning on *fqdn* requires **sudoers** to make DNS lookups which renders **sudo** unusable if DNS stops working (for example if the machine is disconnected from the network). Also note that just like with the hosts file, you must use the “canonical” name as DNS knows it. That is, you may not use a host alias (CNAME entry) due to performance issues and the fact that there is no way to get all aliases from DNS.

This flag is *on* by default.

ignore_audit_errors

Allow commands to be run even if **sudoers** cannot write to the audit log. If enabled, an audit log write failure is not treated as a fatal error. If disabled, a command may only be run after the audit event is successfully written. This flag is only effective on systems for which **sudoers** supports audit logging, including FreeBSD, Linux, macOS, and Solaris. This flag is *on* by default.

ignore_dot

If set, **sudo** will ignore “.” or “” (both denoting current directory) in the PATH environment variable; the PATH itself is not modified. This flag is *off* by default.

ignore_ilog_errors

Allow commands to be run even if **sudoers** cannot write to the I/O log (local or remote). If enabled, an I/O log write failure is not treated as a fatal error. If disabled, the command will be terminated if the I/O log cannot be written to. This flag is *off* by default.

ignore_logfile_errors

Allow commands to be run even if **sudoers** cannot write to the log file. If enabled, a log file write failure is not treated as a fatal error. If disabled, a command may only be run after the log file entry is successfully written. This flag only has an effect when **sudoers** is configured to use file-based logging via the *logfile* setting. This flag is *on*

by default.

`ignore_local_sudoers`

If set via LDAP, parsing of `/etc/sudoers` will be skipped. This is intended for Enterprises that wish to prevent the usage of local sudoers files so that only LDAP is used. This thwarts the efforts of rogue operators who would attempt to add roles to `/etc/sudoers`. When this flag is enabled, `/etc/sudoers` does not even need to exist. Since this flag tells `sudo` how to behave when no specific LDAP entries have been matched, this sudoOption is only meaningful for the `cn=defaults` section. This flag is *off* by default.

`ignore_unknown_defaults`

If set, `sudo` will not produce a warning if it encounters an unknown Defaults entry in the `sudoers` file or an unknown sudoOption in LDAP. This flag is *off* by default.

`insults`

If set, `sudo` will insult users when they enter an incorrect password. This flag is *off* by default.

`log_allowed`

If set, `sudoers` will log commands allowed by the policy to the system audit log (where supported) as well as to syslog and/or a log file. This flag is *on* by default.

This setting is only supported by version 1.8.29 or higher.

`log_denied`

If set, `sudoers` will log commands denied by the policy to the system audit log (where supported) as well as to syslog and/or a log file. This flag is *on* by default.

This setting is only supported by version 1.8.29 or higher.

`log_exit_status`

If set, `sudoers` will log the exit value of commands that are run to syslog and/or a log file. If a command was terminated by a signal, the signal name is logged as well. This flag is *off* by default.

This setting is only supported by version 1.9.8 or higher.

`log_host`

If set, the host name will be included in log entries written to the file configured by the `logfile` setting. This flag is *off* by default.

`log_input`

If set, `sudo` will run the command in a pseudo-terminal and log all user input. If the standard input is not connected to the user's tty, due to I/O redirection or because the command is part of a pipeline, that input is also captured and stored in a separate log file. Anything sent to the standard input will be consumed, regardless of whether or not the command run via `sudo` is actually reading the standard input. This may have unexpected results when using `sudo` in a shell script that expects to process the standard input. For more information about I/O logging, see the **I/O LOG FILES** section. This flag is *off* by default.

`log_output`

If set, `sudo` will run the command in a pseudo-terminal and log all output that is sent to the screen, similar to the `script(1)` command. For more information about I/O logging, see the **I/O LOG FILES** section. This flag is *off* by default.

`log_server_keepalive`

If set, `sudo` will enable the TCP keepalive socket option on the connection to the log server. This enables the periodic transmission of keepalive messages to the server. If the server does not respond to a message, the connection will be closed and the running command will be terminated unless the `ignore_iolog_errors` flag (I/O logging enabled) or the `ignore_log_errors` flag (I/O logging disabled) is set. This flag is *on* by default.

	This setting is only supported by version 1.9.0 or higher.
log_server_verify	If set, the server certificate received during the TLS handshake must be valid and it must contain either the server name (from <i>log_servers</i>) or its IP address. If either of these conditions is not met, the TLS handshake will fail. This flag is <i>on</i> by default.
	This setting is only supported by version 1.9.0 or higher.
log_subcmds	If set, sudoers will log when a command spawns a child process and executes a program using the exec1() , execle() , execlp() , execv() , execve() , execvp() , or execvpe() library functions. For example, if a shell is run by sudo , the individual commands run via the shell will be logged. This flag is <i>off</i> by default.
	The <i>log_subcmds</i> flag uses the same underlying mechanism as the <i>intercept</i> setting. See Preventing shell escapes for more information on what systems support this option and its limitations. This setting is only supported by version 1.9.8 or higher and is incompatible with SELinux RBAC support.
log_year	If set, the four-digit year will be logged in the (non-syslog) sudo log file. This flag is <i>off</i> by default.
long_otp_prompt	When validating with a One Time Password (OTP) scheme such as S/Key or OPIE , a two-line prompt is used to make it easier to cut and paste the challenge to a local window. It's not as pretty as the default but some people find it more convenient. This flag is <i>off</i> by default.
mail_all_cmnds	Send mail to the <i>mailto</i> user every time a user attempts to run a command via sudo (this includes sudoedit). No mail will be sent if the user runs sudo with the -l or -v option unless there is an authentication error and the <i>mail_badpass</i> flag is also set. This flag is <i>off</i> by default.
mail_always	Send mail to the <i>mailto</i> user every time a user runs sudo . This flag is <i>off</i> by default.
mail_badpass	Send mail to the <i>mailto</i> user if the user running sudo does not enter the correct password. If the command the user is attempting to run is not permitted by sudoers and one of the <i>mail_all_cmnds</i> , <i>mail_always</i> , <i>mail_no_host</i> , <i>mail_no_perms</i> or <i>mail_no_user</i> flags are set, this flag will have no effect. This flag is <i>off</i> by default.
mail_no_host	If set, mail will be sent to the <i>mailto</i> user if the invoking user exists in the <i>sudoers</i> file, but is not allowed to run commands on the current host. This flag is <i>off</i> by default.
mail_no_perms	If set, mail will be sent to the <i>mailto</i> user if the invoking user is allowed to use sudo but the command they are trying is not listed in their <i>sudoers</i> file entry or is explicitly denied. This flag is <i>off</i> by default.
mail_no_user	If set, mail will be sent to the <i>mailto</i> user if the invoking user is not in the <i>sudoers</i> file. This flag is <i>on</i> by default.
match_group_by_gid	By default, sudoers will look up each group the user is a member of by group-ID to determine the group name (this is only done once). The resulting list of the user's group names is used when matching groups listed in the <i>sudoers</i> file. This works well on systems where the number of groups listed in the <i>sudoers</i> file is larger than the number of groups a typical user belongs to. On systems where group lookups are slow, where users may belong to a large number of groups, and where the number of groups listed in the <i>sudoers</i> file is relatively small, it may be prohibitively expensive and running commands via sudo may take longer than normal. On such systems it may be faster to use the <i>match_group_by_gid</i> flag to avoid resolving the user's group-IDs to

group names. In this case, ***sudoers*** must look up any group name listed in the *sudoers* file and use the group-ID instead of the group name when determining whether the user is a member of the group.

Note that if *match_group_by_gid* is enabled, group database lookups performed by ***sudoers*** will be keyed by group name as opposed to group-ID. On systems where there are multiple sources for the group database, it is possible to have conflicting group names or group-IDs in the local */etc/group* file and the remote group database. On such systems, enabling or disabling *match_group_by_gid* can be used to choose whether group database queries are performed by name (enabled) or ID (disabled), which may aid in working around group entry conflicts.

The *match_group_by_gid* flag has no effect when *sudoers* data is stored in LDAP. This flag is *off* by default.

This setting is only supported by version 1.8.18 or higher.

intercept

If set, all commands run via ***sudo*** will behave as if the *INTERCEPT* tag has been set, unless overridden by an *NOINTERCEPT* tag. See the description of *INTERCEPT* and *NOINTERCEPT* above as well as the **Preventing shell escapes** section at the end of this manual. This flag is *off* by default.

This setting is only supported by version 1.9.8 or higher and is incompatible with SELinux RBAC support.

intercept_allow_setid

On most systems, the dynamic loader will ignore *LD_PRELOAD* (or the equivalent) when running set-user-ID and set-group-ID programs, effectively disabling intercept mode. To prevent this from happening, ***sudoers*** will not permit a set-user-ID or set-group-ID program to be run in intercept mode unless *intercept_allow_setid* is set. This flag has no effect unless the *intercept* flag is enabled or the *INTERCEPT* tag has been set for the command. This flag is *on* by default.

This setting is only supported by version 1.9.8 or higher.

intercept_authenticate

If set, commands run by an intercepted process must be authenticated when the user's time stamp is not current. For example, if a shell is run with *intercept* enabled, as soon as the invoking user's time stamp is out of date, subsequent commands will need to be authenticated. This flag has no effect unless the *intercept* flag is enabled or the *INTERCEPT* tag has been set for the command. This flag is *off* by default.

This setting is only supported by version 1.9.8 or higher.

netgroup_tuple

If set, netgroup lookups will be performed using the full netgroup tuple: host name, user name, and domain (if one is set). Historically, ***sudo*** only matched the user name and domain for netgroups used in a *User_List* and only matched the host name and domain for netgroups used in a *Host_List*. This flag is *off* by default.

noexec

If set, all commands run via ***sudo*** will behave as if the *NOEXEC* tag has been set, unless overridden by an *EXEC* tag. See the description of *EXEC* and *NOEXEC* above as well as the **Preventing shell escapes** section at the end of this manual. This flag is *off* by default.

pam_acct_mgmt

On systems that use PAM for authentication, ***sudo*** will perform PAM account validation for the invoking user by default. The actual checks performed depend on which PAM modules are configured. If enabled, account validation will be performed regardless of whether or not a password is required. This flag is *on* by default.

	This setting is only supported by version 1.8.28 or higher.
pam_rhost	On systems that use PAM for authentication, sudo will set the PAM remote host value to the name of the local host when the <i>pam_rhost</i> flag is enabled. On Linux systems, enabling <i>pam_rhost</i> may result in DNS lookups of the local host name when PAM is initialized. On Solaris versions prior to Solaris 8, <i>pam_rhost</i> must be enabled if <i>pam_ruser</i> is also enabled to avoid a crash in the Solaris PAM implementation. This flag is <i>off</i> by default on systems other than Solaris.
	This setting is only supported by version 1.9.0 or higher.
pam_ruser	On systems that use PAM for authentication, sudo will set the PAM remote user value to the name of the user that invoked sudo when the <i>pam_ruser</i> flag is enabled. This flag is <i>on</i> by default. This setting is only supported by version 1.9.0 or higher.
pam_session	On systems that use PAM for authentication, sudo will create a new PAM session for the command to be run in. Unless sudo is given the -i or -s options, PAM session modules are run with the “silent” flag enabled. This prevents last login information from being displayed for every command on some systems. Disabling <i>pam_session</i> may be needed on older PAM implementations or on operating systems where opening a PAM session changes the utmp or wtmp files. If PAM session support is disabled, resource limits may not be updated for the command being run. If <i>pam_session</i> , <i>pam_setcred</i> , and <i>use_pty</i> are disabled, <i>log_servers</i> has not been set and I/O logging has not been configured, sudo will execute the command directly instead of running it as a child process. This flag is <i>on</i> by default. This setting is only supported by version 1.8.7 or higher.
pam_setcred	On systems that use PAM for authentication, sudo will attempt to establish credentials for the target user by default, if supported by the underlying authentication system. One example of a credential is a Kerberos ticket. If <i>pam_session</i> , <i>pam_setcred</i> , and <i>use_pty</i> are disabled, <i>log_servers</i> has not been set and I/O logging has not been configured, sudo will execute the command directly instead of running it as a child process. This flag is <i>on</i> by default. This setting is only supported by version 1.8.8 or higher.
passprompt_override	If set, the prompt specified by <i>passprompt</i> or the SUDO_PROMPT environment variable will always be used and will replace the prompt provided by a PAM module or other authentication method. This flag is <i>off</i> by default.
path_info	Normally, sudo will tell the user when a command could not be found in their PATH environment variable. Some sites may wish to disable this as it could be used to gather information on the location of executables that the normal user does not have access to. The disadvantage is that if the executable is simply not in the user’s PATH, sudo will tell the user that they are not allowed to run it, which can be confusing. This flag is <i>on</i> by default.
preserve_groups	By default, sudo will initialize the group vector to the list of groups the target user is in. When <i>preserve_groups</i> is set, the user’s existing group vector is left unaltered. The real and effective group-IDs, however, are still set to match the target user. This flag is <i>off</i> by default.

pwfeedback	By default, sudo reads the password like most other Unix programs, by turning off echo until the user hits the return (or enter) key. Some users become confused by this as it appears to them that sudo has hung at this point. When <i>pwfeedback</i> is set, sudo will provide visual feedback when the user presses a key. Note that this does have a security impact as an onlooker may be able to determine the length of the password being entered. This flag is <i>off</i> by default.
requiretty	If set, sudo will only run when the user is logged in to a real tty. When this flag is set, sudo can only be run from a login session and not via other means such as cron(8) or cgi-bin scripts. This flag is <i>off</i> by default.
root_sudo	If set, root is allowed to run sudo too. Disabling this prevents users from “chaining” sudo commands to get a root shell by doing something like “ sudo sudo /bin/sh ”. Note, however, that turning off <i>root_sudo</i> will also prevent root from running sudoedit . Disabling <i>root_sudo</i> provides no real additional security; it exists purely for historical reasons. This flag is <i>on</i> by default.
rootpw	If set, sudo will prompt for the root password instead of the password of the invoking user when running a command or editing a file. This flag is <i>off</i> by default.
runas_allow_unknown_id	If enabled, allow matching of runas user and group IDs that are not present in the password or group databases. In addition to explicitly matching unknown user or group IDs in a Runas_List, this option also allows the ALL alias to match unknown IDs. This flag is <i>off</i> by default. This setting is only supported by version 1.8.30 or higher. Older versions of sudo always allowed matching of unknown user and group IDs.
runas_check_shell	If enabled, sudo will only run commands as a user whose shell appears in the <i>/etc/shells</i> file, even if the invoking user’s Runas_List would otherwise permit it. If no <i>/etc/shells</i> file is present, a system-dependent list of built-in default shells is used. On many operating systems, system users such as “bin”, do not have a valid shell and this flag can be used to prevent commands from being run as those users. This flag is <i>off</i> by default. This setting is only supported by version 1.8.30 or higher.
runaspw	If set, sudo will prompt for the password of the user defined by the <i>runas_default</i> option (defaults to <i>root</i>) instead of the password of the invoking user when running a command or editing a file. This flag is <i>off</i> by default.
selinux	If enabled, the user may specify an SELinux role and/or type to use when running the command, as permitted by the SELinux policy. If SELinux is disabled on the system, this flag has no effect. This flag is <i>on</i> by default.
set_home	If enabled and sudo is invoked with the -s option, the HOME environment variable will be set to the home directory of the target user (which is the root user unless the -u option is used). This flag is largely obsolete and has no effect unless the <i>env_reset</i> flag has been disabled or HOME is present in the <i>env_keep</i> list, both of which are strongly discouraged. This flag is <i>off</i> by default.
set_logname	Normally, sudo will set the LOGNAME and USER environment variables to the name of the target user (usually root unless the -u option is given). However, since some programs (including the RCS revision control system) use LOGNAME to determine the real identity of the user, it may be desirable to change this behavior. This can be done by negating the <i>set_logname</i> option. Note that <i>set_logname</i> will have no effect if the

	<i>env_reset</i> option has not been disabled and the <i>env_keep</i> list contains LOGNAME or USER. This flag is <i>on</i> by default.
set_utmp	When enabled, sudo will create an entry in the utmp (or utmpx) file when a pseudo-terminal is allocated. A pseudo-terminal is allocated by sudo when it is running in a terminal and one or more of the <i>log_input</i> , <i>log_output</i> , or <i>use_pty</i> flags is enabled. By default, the new entry will be a copy of the user's existing utmp entry (if any), with the tty, time, type, and pid fields updated. This flag is <i>on</i> by default.
setenv	Allow the user to disable the <i>env_reset</i> option from the command line via the -E option. Additionally, environment variables set via the command line are not subject to the restrictions imposed by <i>env_check</i> , <i>env_delete</i> , or <i>env_keep</i> . As such, only trusted users should be allowed to set variables in this manner. This flag is <i>off</i> by default.
shell_noargs	If set and sudo is invoked with no arguments it acts as if the -s option had been given. That is, it runs a shell as root (the shell is determined by the <i>SHELL</i> environment variable if it is set, falling back on the shell listed in the invoking user's /etc/passwd entry if not). This flag is <i>off</i> by default.
stay_setuid	Normally, when sudo executes a command the real and effective user-IDs are set to the target user (root by default). This option changes that behavior such that the real user-ID is left as the invoking user's user-ID. In other words, this makes sudo act as a set-user-ID wrapper. This can be useful on systems that disable some potentially dangerous functionality when a program is run set-user-ID. This option is only effective on systems that support either the <i>setreuid(2)</i> or <i>setresuid(2)</i> system call. This flag is <i>off</i> by default.
sudoedit_checkdir	If set, sudoedit will check all directory components of the path to be edited for writability by the invoking user. Symbolic links will not be followed in writable directories and sudoedit will refuse to edit a file located in a writable directory. These restrictions are not enforced when sudoedit is run by root. On some systems, if all directory components of the path to be edited are not readable by the target user, sudoedit will be unable to edit the file. This flag is <i>on</i> by default. This setting was first introduced in version 1.8.15 but initially suffered from a race condition. The check for symbolic links in writable intermediate directories was added in version 1.8.16.
sudoedit_follow	By default, sudoedit will not follow symbolic links when opening files. The <i>sudoedit_follow</i> option can be enabled to allow sudoedit to open symbolic links. It may be overridden on a per-command basis by the <i>FOLLOW</i> and <i>NOFOLLOW</i> tags. This flag is <i>off</i> by default. This setting is only supported by version 1.8.15 or higher.
syslog_pid	When logging via <i>syslog(3)</i> , include the process ID in the log entry. This flag is <i>off</i> by default.
	This setting is only supported by version 1.8.21 or higher.
targetpw	If set, sudo will prompt for the password of the user specified by the -u option (defaults to <i>root</i>) instead of the password of the invoking user when running a command or editing a file. Note that this flag precludes the use of a user-ID not listed in the passwd database as an argument to the -u option. This flag is <i>off</i> by default.
tty_tickets	If set, users must authenticate on a per-tty basis. With this flag enabled, sudo will use a separate record in the time stamp file for each terminal. If disabled, a single record is used for all login sessions.

	This option has been superseded by the <i>timestamp_type</i> option.
umask_override	If set, sudo will set the umask as specified in the <i>sudoers</i> file without modification. This makes it possible to specify a umask in the <i>sudoers</i> file that is more permissive than the user's own umask and matches historical behavior. If <i>umask_override</i> is not set, sudo will set the umask to be the union of the user's umask and what is specified in <i>sudoers</i> . This flag is <i>off</i> by default.
use_netgroups	If set, netgroups (prefixed with '+'), may be used in place of a user or host. For LDAP-based sudoers, netgroup support requires an expensive sub-string match on the server unless the NETGROUP_BASE directive is present in the <i>/etc/ldap.conf</i> file. If netgroups are not needed, this option can be disabled to reduce the load on the LDAP server. This flag is <i>on</i> by default.
use_pty	If set, and sudo is running in a terminal, the command will be run in a pseudo-terminal (even if no I/O logging is being done). If the sudo process is not attached to a terminal, <i>use_pty</i> has no effect.
	A malicious program run under sudo may be capable of injecting commands into the user's terminal or running a background process that retains access to the user's terminal device even after the main program has finished executing. By running the command in a separate pseudo-terminal, this attack is no longer possible. This flag is <i>off</i> by default.
user_command_timeouts	If set, the user may specify a timeout on the command line. If the timeout expires before the command has exited, the command will be terminated. If a timeout is specified both in the <i>sudoers</i> file and on the command line, the smaller of the two timeouts will be used. See the Timeout_Spec section for a description of the timeout syntax. This flag is <i>off</i> by default.
	This setting is only supported by version 1.8.20 or higher.
utmp_runas	If set, sudo will store the name of the runas user when updating the utmp (or utmpx) file. By default, sudo stores the name of the invoking user. This flag is <i>off</i> by default.
visiblepw	By default, sudo will refuse to run if the user must enter a password but it is not possible to disable echo on the terminal. If the <i>visiblepw</i> flag is set, sudo will prompt for a password even when it would be visible on the screen. This makes it possible to run things like "ssh somehost sudo ls" since by default, ssh(1) does not allocate a tty when running a command. This flag is <i>off</i> by default.
Integers:	
closefrom	Before it executes a command, sudo will close all open file descriptors other than standard input, standard output, and standard error (file descriptors 0-2). The <i>closefrom</i> option can be used to specify a different file descriptor at which to start closing. The default is 3.
command_timeout	The maximum amount of time a command is allowed to run before it is terminated. See the Timeout_Spec section for a description of the timeout syntax.
	This setting is only supported by version 1.8.20 or higher.
log_server_timeout	The maximum amount of time to wait when connecting to a log server or waiting for a server response. See the Timeout_Spec section for a description of the timeout syntax. The default value is 30 seconds.

	This setting is only supported by version 1.9.0 or higher.
maxseq	The maximum sequence number that will be substituted for the “%{seq}” escape in the I/O log file (see the <i>iolog_dir</i> description below for more information). While the value substituted for “%{seq}” is in base 36, <i>maxseq</i> itself should be expressed in decimal. Values larger than 2176782336 (which corresponds to the base 36 sequence number “ZZZZZZ”) will be silently truncated to 2176782336. The default value is 2176782336.
	Once the local sequence number reaches the value of <i>maxseq</i> , it will “roll over” to zero, after which sudoers will truncate and re-use any existing I/O log path names.
	This setting is only supported by version 1.8.7 or higher.
passwd_tries	The number of tries a user gets to enter his/her password before sudo logs the failure and exits. The default is 3.
syslog_maxlen	On many systems, <i>syslog</i> (3) has a relatively small log buffer. IETF RFC 5424 states that syslog servers must support messages of at least 480 bytes and should support messages up to 2048 bytes. By default, sudoers creates log messages up to 980 bytes which corresponds to the historic BSD syslog implementation which used a 1024 byte buffer to store the message, date, hostname, and program name. To prevent syslog messages from being truncated, sudoers will split up log messages that are larger than <i>syslog_maxlen</i> bytes. When a message is split, additional parts will include the string “(command continued)” after the user name and before the continued command line arguments.
	This setting is only supported by version 1.8.19 or higher.
Integers that can be used in a boolean context:	
loglinelen	Number of characters per line for the file log. This value is used to decide when to wrap lines for nicer log files. This has no effect on the syslog log file, only the file log. The default is 80 (use 0 or negate the option to disable word wrap).
passwd_timeout	Number of minutes before the sudo password prompt times out, or 0 for no timeout. The timeout may include a fractional component if minute granularity is insufficient, for example 2.5. The default is 0.
timestamp_timeout	Number of minutes that can elapse before sudo will ask for a passwd again. The timeout may include a fractional component if minute granularity is insufficient, for example 2.5. The default is 15. Set this to 0 to always prompt for a password. If set to a value less than 0 the user’s time stamp will not expire until the system is rebooted. This can be used to allow users to create or delete their own time stamps via “ sudo -v ” and “ sudo -k ” respectively.
umask	File mode creation mask to use when running the command. Negate this option or set it to 0777 to prevent sudoers from changing the umask. Unless the <i>umask_override</i> flag is set, the actual umask will be the union of the user’s umask and the value of the <i>umask</i> setting, which defaults to 0022. This guarantees that sudo never lowers the umask when running a command.
	If <i>umask</i> is explicitly set in <i>sudoers</i> , it will override any umask setting in PAM or login.conf. If <i>umask</i> is not set in <i>sudoers</i> , the umask specified by PAM or login.conf will take precedence. The umask setting in PAM is not used for sudoedit , which does not create a new PAM session.

Strings:

authfail_message	Message that is displayed after a user fails to authenticate. The message may include the ‘%d’ escape which will expand to the number of failed password attempts. If set, it overrides the default message, %d incorrect password attempt(s).
badpass_message	Message that is displayed if a user enters an incorrect password. The default is Sorry, try again.unless insults are enabled.
editor	A colon (‘:’) separated list of editors path names used by sudoedit and visudo . For sudoedit , this list is used to find an editor when none of the SUDO_EDITOR, VISUAL or EDITOR environment variables are set to an editor that exists and is executable. For visudo , it is used as a white list of allowed editors; visudo will choose the editor that matches the user’s SUDO_EDITOR, VISUAL or EDITOR environment variable if possible, or the first editor in the list that exists and is executable if not. Unless invoked as sudoedit , sudo does not preserve the SUDO_EDITOR, VISUAL or EDITOR environment variables unless they are present in the <i>env_keep</i> list or the <i>env_reset</i> option is disabled. The default is /usr/bin/editor.
ilog_dir	The top-level directory to use when constructing the path name for the input/output log directory. Only used if the <i>log_input</i> or <i>log_output</i> options are enabled or when the LOG_INPUT or LOG_OUTPUT tags are present for a command. The session sequence number, if any, is stored in the directory. The default is /var/log/sudo-io. The following percent (‘%’) escape sequences are supported: %{seq} expanded to a monotonically increasing base-36 sequence number, such as 0100A5, where every two digits are used to form a new directory, e.g., 01/00/A5 %{user} expanded to the invoking user’s login name %{group} expanded to the name of the invoking user’s real group-ID %{runas_user} expanded to the login name of the user the command will be run as (e.g., root) %{runas_group} expanded to the group name of the user the command will be run as (e.g., wheel) %{hostname} expanded to the local host name without the domain name %{command} expanded to the base name of the command being run In addition, any escape sequences supported by the system’s strftime(3) function will be expanded. To include a literal ‘%’ character, the string ‘%%’ should be used. ilog_file The path name, relative to <i>ilog_dir</i> , in which to store input/output logs when the <i>log_input</i> or <i>log_output</i> options are enabled or when the LOG_INPUT or LOG_OUTPUT tags are present for a command. Note that <i>ilog_file</i> may contain directory components. The default is “%{seq}”.

	See the <i>iolog_dir</i> option above for a list of supported percent ('%') escape sequences. In addition to the escape sequences, path names that end in six or more Xs will have the Xs replaced with a unique combination of digits and letters, similar to the <code>mktemp(3)</code> function.
<i>iolog_flush</i>	If the path created by concatenating <i>iolog_dir</i> and <i>iolog_file</i> already exists, the existing I/O log file will be truncated and overwritten unless <i>iolog_file</i> ends in six or more Xs.
<i>iolog_group</i>	If set, sudo will flush I/O log data to disk after each write instead of buffering it. This makes it possible to view the logs in real-time as the program is executing but may significantly reduce the effectiveness of I/O log compression. This flag is <i>off</i> by default.
	This setting is only supported by version 1.8.20 or higher.
<i>iolog_mode</i>	The group name to look up when setting the group-ID on new I/O log files and directories. If <i>iolo g_group</i> is not set, the primary group-ID of the user specified by <i>iolog_user</i> is used. If neither <i>iolog_group</i> nor <i>iolog_user</i> are set, I/O log files and directories are created with group-ID 0.
	This setting is only supported by version 1.8.19 or higher.
<i>iolog_user</i>	The file mode to use when creating I/O log files. Mode bits for read and write permissions for owner, group, or other are honored, everything else is ignored. The file permissions will always include the owner read and write bits, even if they are not present in the specified mode. When creating I/O log directories, search (execute) bits are added to match the read and write bits specified by <i>iolog_mode</i> . Defaults to 0600 (read and write by user only).
	This setting is only supported by version 1.8.19 or higher.
<i>lecture_status_dir</i>	The user name to look up when setting the user and group-IDs on new I/O log files and directories. If <i>iolo g_group</i> is set, it will be used instead of the user's primary group-ID. By default, I/O log files and directories are created with user and group-ID 0.
	This setting can be useful when the I/O logs are stored on a Network File System (NFS) share. Having a dedicated user own the I/O log files means that sudoers does not write to the log files as user-ID 0, which is usually not permitted by NFS.
	This setting is only supported by version 1.8.19 or higher.
<i>log_server_cabundle</i>	The directory in which sudo stores per-user lecture status files. Once a user has received the lecture, a zero-length file is created in this directory so that sudo will not lecture the user again. This directory should not be cleared when the system reboots. The default is <code>/var/lib/sudo/lectured</code> .
	The path to a certificate authority bundle file, in PEM format, to use instead of the system's default certificate authority database when authenticating the log server. The default is to use the system's default certificate authority database. This setting has no effect unless <i>log_servers</i> is set and the remote log server is secured with TLS.
	This setting is only supported by version 1.9.0 or higher.
<i>log_server_peer_cert</i>	The path to the sudo client's certificate file, in PEM format. This setting is required when the remote log server is secured with TLS and client certificate validation is enabled. For sudo_logsvrd , client certificate validation is controlled by the <i>tls_checkpeer</i> option, which defaults to <i>false</i> .

This setting is only supported by version 1.9.0 or higher.

log_server_peer_key

The path to the **sudo** client's private key file, in PEM format. This setting is required when the remote log server is secured with TLS and client certificate validation is enabled. For **sudo_logsvrd**, client certificate validation is controlled by the *tls_checkpeer* option, which defaults to *false*.

This setting is only supported by version 1.9.0 or higher.

mailsub

Subject of the mail sent to the *mailto* user. The escape%*h* will expand to the host name of the machine. Default is “*** SECURITY information for %*h* ***”.

noexec_file

As of **sudo** version 1.8.1 this option is no longer supported. The path to the noexec file should now be set in the **sudo.conf(5)** file.

pam_askpass_service

On systems that use PAM for authentication, this is the service name used when the **-A** option is specified. The default value is either “@pam_service@” or “**sudo-i**”, depending on whether or not the **-i** option is also specified. See the description of **pam_service** for more information.

This setting is only supported by version 1.9.9 or higher.

pam_login_service

On systems that use PAM for authentication, this is the service name used when the **-i** option is specified. The default value is “**sudo-i**”. See the description of **pam_service** for more information.

This setting is only supported by version 1.8.8 or higher.

pam_service

On systems that use PAM for authentication, the service name specifies the PAM policy to apply. This usually corresponds to an entry in the **pam.conf** file or a file in the **/etc/pam.d** directory. The default value is “**sudo**”.

This setting is only supported by version 1.8.8 or higher.

passprompt

The default prompt to use when asking for a password; can be overridden via the **-p** option or the **SUDO_PROMPT** environment variable. The following percent ('%') escape sequences are supported:

%H expanded to the local host name including the domain name (only if the machine's host name is fully qualified or the *fqdn* option is set)

%h expanded to the local host name without the domain name

%p expanded to the user whose password is being asked for (respects the *rootpw*, *targetpw* and *runaspw* flags in **sudoers**)

%U expanded to the login name of the user the command will be run as (defaults to root)

%u expanded to the invoking user's login name

%% two consecutive % characters are collapsed into a single % character

On systems that use PAM for authentication, *passprompt* will only be used if the prompt provided by the PAM module matches the string “Password: ” or “username's Password: ”. This ensures that the *passprompt* setting does not interfere with challenge-response style authentication. The *passprompt_override* flag can be used to change this behavior.

	The default value is “[sudo] password for %p: ”.
role	The default SELinux role to use when constructing a new security context to run the command. The default role may be overridden on a per-command basis in the <i>sudoers</i> file or via command line options. This option is only available when sudo is built with SELinux support.
runas_default	The default user to run commands as if the -u option is not specified on the command line. This defaults to <code>root</code> .
sudoers_locale	Locale to use when parsing the sudoers file, logging commands, and sending email. Note that changing the locale may affect how sudoers is interpreted. Defaults to “C”.
timestamp_type	sudoers uses per-user time stamp files for credential caching. The <i>timestamp_type</i> option can be used to specify the type of time stamp record used. It has the following possible values:
global	A single time stamp record is used for all of a user’s login sessions, regardless of the terminal or parent process ID. An additional record is used to serialize password prompts when sudo is used multiple times in a pipeline, but this does not affect authentication.
ppid	A single time stamp record is used for all processes with the same parent process ID (usually the shell). Commands run from the same shell (or other common parent process) will not require a password for <i>timestamp_timeout</i> minutes (15 by default). Commands run via sudo with a different parent process ID, for example from a shell script, will be authenticated separately.
tty	One time stamp record is used for each terminal, which means that a user’s login sessions are authenticated separately. If no terminal is present, the behavior is the same as <i>ppid</i> . Commands run from the same terminal will not require a password for <i>timestamp_timeout</i> minutes (15 by default).
kernel	The time stamp is stored in the kernel as an attribute of the terminal device. If no terminal is present, the behavior is the same as <i>ppid</i> . Negative <i>timestamp_timeout</i> values are not supported and positive values are limited to a maximum of 60 minutes. This is currently only supported on OpenBSD.
	The default value is <code>tty</code> .
	This setting is only supported by version 1.8.21 or higher.
timestampdir	The directory in which sudo stores its time stamp files. This directory should be cleared when the system reboots. The default is <code>/run/sudo/ts</code> .
timestampowner	The owner of the lecture status directory, time stamp directory and all files stored therein. The default is <code>root</code> .
type	The default SELinux type to use when constructing a new security context to run the command. The default type may be overridden on a per-command basis in the <i>sudoers</i> file or via command line options. This option is only available when sudo is built with SELinux support.

Strings that can be used in a boolean context:

admin_flag	The <i>admin_flag</i> option specifies the path to a file that is created the first time a user that is a member of the <i>sudo</i> or <i>admin</i> groups runs sudo . Only available if sudo is configured with the <code>--enable-admin-flag</code> option. The default value is <code>~/.sudo_as_admin_successful</code> .
------------	---

env_file	The <i>env_file</i> option specifies the fully qualified path to a file containing variables to be set in the environment of the program being run. Entries in this file should either be of the form “VARIABLE=value” or “export VARIABLE=value”. The value may optionally be enclosed in single or double quotes. Variables in this file are only added if the variable does not already exist in the environment. This file is considered to be part of the security policy, its contents are not subject to other sudo environment restrictions such as <i>env_keep</i> and <i>env_check</i> .
exempt_group	Users in this group are exempt from password and PATH requirements. The group name specified should not include a % prefix. This is not set by default.
fdexec	Determines whether sudo will execute a command by its path or by an open file descriptor. It has the following possible values: always Always execute by file descriptor. never Never execute by file descriptor. digest_only Only execute by file descriptor if the command has an associated digest in the <i>sudoers</i> file. The default value is <i>digest_only</i> . This avoids a time of check versus time of use race condition when the command is located in a directory writable by the invoking user. Note that <i>fdexec</i> will change the first element of the argument vector for scripts (\$0 in the shell) due to the way the kernel runs script interpreters. Instead of being a normal path, it will refer to a file descriptor. For example, /dev/fd/4 on Solaris and /proc/self/fd/4 on Linux. A workaround is to use the SUDO_COMMAND environment variable instead. The <i>fdexec</i> setting is only used when the command is matched by path name. It has no effect if the command is matched by the built-in ALL alias. This setting is only supported by version 1.8.20 or higher. If the operating system does not support the fexecve() system call, this setting has no effect.
group_plugin	A string containing a sudoers group plugin with optional arguments. The string should consist of the plugin path, either fully-qualified or relative to the /usr/libexec/sudo directory, followed by any configuration arguments the plugin requires. These arguments (if any) will be passed to the plugin’s initialization function. If arguments are present, the string must be enclosed in double quotes (""). For more information see GROUP PROVIDER PLUGINS .
lecture	This option controls when a short lecture will be printed along with the password prompt. It has the following possible values: always Always lecture the user. never Never lecture the user. once Only lecture the user the first time they run sudo . If no value is specified, a value of <i>once</i> is implied. Negating the option results in a value of <i>never</i> being used. The default value is <i>never</i> .
lecture_file	Path to a file containing an alternate sudo lecture that will be used in place of the standard lecture if the named file exists. By default, sudo uses a built-in lecture.

listpw	This option controls when a password will be required when a user runs sudo with the -l option. It has the following possible values:
all	All the user's <i>sudoers</i> file entries for the current host must have the NOPASSWD flag set to avoid entering a password.
always	The user must always enter a password to use the -l option.
any	At least one of the user's <i>sudoers</i> file entries for the current host must have the NOPASSWD flag set to avoid entering a password.
never	The user need never enter a password to use the -l option.
	If no value is specified, a value of <i>any</i> is implied. Negating the option results in a value of <i>never</i> being used. The default value is <i>any</i> .
log_format	The event log format. Supported log formats are:
json	Logs in JSON format. JSON log entries contain the full user details as well as the execution environment if the command was allowed. Due to limitations of the protocol, JSON events sent via <i>syslog</i> may be truncated.
sudo	Traditional sudo-style logs, see LOG FORMAT for a description of the log file format.
	This setting affects logs sent via <i>syslog(3)</i> as well as the file specified by the <i>logfile</i> setting, if any. The default value is <i>sudo</i> .
logfile	Path to the sudo log file (not the syslog log file). Setting a path turns on logging to a file; negating this option turns it off. By default, sudo logs via syslog.
mailerflags	Flags to use when invoking mailer. Defaults to -t .
mailerpath	Path to mail program used to send warning mail. Defaults to the path to sendmail found at configure time.
mailfrom	Address to use for the "from" address when sending warning and error mail. The address should be enclosed in double quotes ("") to protect against sudo interpreting the @ sign. Defaults to the name of the user running sudo .
mailto	Address to send warning and error mail to. The address should be enclosed in double quotes ("") to protect against sudo interpreting the @ sign. Defaults to root .
rlimit_as	The maximum size to which the process's address space may grow (in bytes), if supported by the operating system. See Resource limits for more information.
rlimit_core	The largest size core dump file that may be created (in bytes). See Resource limits for more information. Defaults to 0 (no core dump created).
rlimit_cpu	The maximum amount of CPU time that the process may use (in seconds). See Resource limits for more information.
rlimit_data	The maximum size of the data segment for the process (in bytes). See Resource limits for more information.
rlimit_fsize	The largest size file that the process may create (in bytes). See Resource limits for more information.
rlimit_locks	The maximum number of locks that the process may establish, if supported by the operating system. See Resource limits for more information.

rlimit_memlock

The maximum size that the process may lock in memory (in bytes), if supported by the operating system. See **Resource limits** for more information.

rlimit_nofile

The maximum number of files that the process may have open. See **Resource limits** for more information.

rlimit_nproc

The maximum number of processes that the user may run simultaneously. See **Resource limits** for more information.

rlimit_rss

The maximum size to which the process's resident set size may grow (in bytes). See **Resource limits** for more information.

rlimit_stack

The maximum size to which the process's stack may grow (in bytes). See **Resource limits** for more information.

restricted_env_file

The *restricted_env_file* option specifies the fully qualified path to a file containing variables to be set in the environment of the program being run. Entries in this file should either be of the form “`VARIABLE=value`” or “`export VARIABLE=value`”. The value may optionally be enclosed in single or double quotes. Variables in this file are only added if the variable does not already exist in the environment. Unlike *env_file*, the file's contents are not trusted and are processed in a manner similar to that of the invoking user's environment. If *env_reset* is enabled, variables in the file will only be added if they are matched by either the *env_check* or *env_keep* list. If *env_reset* is disabled, variables in the file are added as long as they are not matched by the *env_delete* list. In either case, the contents of *restricted_env_file* are processed before the contents of *env_file*.

runchroot

If set, **sudo** will use this value for the root directory when running a command. The special value “`*`” will allow the user to specify the root directory via **sudo**'s **-R** option. See the **Chroot_Spec** section for more details.

It is only possible to use *runchroot* as a command-specific Defaults setting if the command exists with the same path both inside and outside the chroot jail. This restriction does not apply to generic, host, or user-based Defaults settings or to a *Cmnd_Spec* that includes a *Chroot_Spec*.

This setting is only supported by version 1.9.3 or higher.

runcwd

If set, **sudo** will use this value for the working directory when running a command. The special value “`*`” will allow the user to specify the working directory via **sudo**'s **-D** option. See the **Chdir_Spec** section for more details.

This setting is only supported by version 1.9.3 or higher.

secure_path

If set, **sudo** will use this value in place of the user's PATH environment variable. This option can be used to reset the PATH to a known good value that contains directories for system administrator commands such as `/usr/sbin`.

Users in the group specified by the *exempt_group* option are not affected by *secure_path*. This option is not set by default.

syslog

Syslog facility if syslog is being used for logging (negate to disable syslog logging). Defaults to authpriv.

The following syslog facilities are supported: **authpriv** (if your OS supports it), **auth**, **daemon**, **user**, **local0**, **local1**, **local2**, **local3**, **local4**, **local5**, **local6**, and **local7**.

syslog_badpri Syslog priority to use when the user is not allowed to run a command or when authentication is unsuccessful. Defaults to **alert**.

The following syslog priorities are supported: **alert**, **crit**, **debug**, **emerg**, **err**, **info**, **notice**, **warning**, and **none**. Negating the option or setting it to a value of **none** will disable logging of unsuccessful commands.

syslog_goodpri

Syslog priority to use when the user is allowed to run a command and authentication is successful. Defaults to **notice**.

See *syslog_badpri* for the list of supported syslog priorities. Negating the option or setting it to a value of **none** will disable logging of successful commands.

verifypw

This option controls when a password will be required when a user runs **sudo** with the **-v** option. It has the following possible values:

- all** All the user's *sudoers* file entries for the current host must have the NOPASSWD flag set to avoid entering a password.
- always** The user must always enter a password to use the **-v** option.
- any** At least one of the user's *sudoers* file entries for the current host must have the NOPASSWD flag set to avoid entering a password.
- never** The user need never enter a password to use the **-v** option.

If no value is specified, a value of *all* is implied. Negating the option results in a value of *never* being used. The default value is *all*.

Lists that can be used in a boolean context:

env_check

Environment variables to be removed from the user's environment unless they are considered "safe". For all variables except TZ, "safe" means that the variable's value does not contain any '%' or '/' characters. This can be used to guard against printf-style format vulnerabilities in poorly-written programs. The TZ variable is considered unsafe if any of the following are true:

- It consists of a fully-qualified path name, optionally prefixed with a colon (':'), that does not match the location of the *zoneinfo* directory.
- It contains a . . path element.
- It contains white space or non-printable characters.
- It is longer than the value of PATH_MAX.

The argument may be a double-quoted, space-separated list or a single value without double-quotes. The list can be replaced, added to, deleted from, or disabled by using the =, +=, -=, and ! operators respectively. Regardless of whether the *env_reset* option is enabled or disabled, variables specified by *env_check* will be preserved in the environment if they pass the aforementioned check. The global list of environment variables to check is displayed when **sudo** is run by root with the **-V** option.

env_delete

Environment variables to be removed from the user's environment when the *env_reset* option is not in effect. The argument may be a double-quoted, space-separated list or a single value without double-quotes. The list can be replaced, added to, deleted from, or disabled by using the =, +=, -=, and ! operators respectively. The global list of environment variables to remove is displayed when **sudo** is run by root with the **-V** option. Note that many operating systems will remove potentially dangerous variables from the

	environment of any set-user-ID process (such as sudo).
env_keep	Environment variables to be preserved in the user's environment when the <i>env_reset</i> option is in effect. This allows fine-grained control over the environment sudo -spawned processes will receive. The argument may be a double-quoted, space-separated list or a single value without double-quotes. The list can be replaced, added to, deleted from, or disabled by using the <code>=</code> , <code>+ =</code> , <code>- =</code> , and <code>!</code> operators respectively. The global list of variables to keep is displayed when sudo is run by root with the <code>-V</code> option.
	Preserving the <code>HOME</code> environment variable has security implications since many programs use it when searching for configuration or data files. Adding <code>HOME</code> to <i>env_keep</i> may enable a user to run unrestricted commands via sudo and is strongly discouraged. Users wishing to edit files with sudo should run sudoedit (or sudo -e) to get their accustomed editor configuration instead of invoking the editor directly.
log_servers	A list of one or more servers to use for remote event and I/O log storage, separated by white space. Log servers must be running sudo_logsrvd or another service that implements the protocol described by sudo_logsrv.proto(5) . Server addresses should be of the form “host[:port][(tls)]”. The host portion may be a host name, an IPv4 address, or an IPv6 address in square brackets. If the optional <i>tls</i> flag is present, the connection will be secured with Transport Layer Security (TLS) version 1.2 or 1.3. Versions of TLS prior to 1.2 are not supported. If a port is specified, it may either be a port number or a well-known service name as defined by the system service name database. If no port is specified, port 30343 will be used for plaintext connections and port 30344 will be used for TLS connections. When <i>log_servers</i> is set, event log data will be logged both locally (see the <i>syslog</i> and <i>log_file</i> settings) as well as remotely, but I/O log data will only be logged remotely. If multiple hosts are specified, they will be attempted in reverse order. If no log servers are available, the user will not be able to run a command unless either the <i>ignore_iolog_errors</i> flag (I/O logging enabled) or the <i>ignore_log_errors</i> flag (I/O logging disabled) is set. Likewise, if the connection to the log server is interrupted while sudo is running, the command will be terminated unless the <i>ignore_iolog_errors</i> flag (I/O logging enabled) or the <i>ignore_log_errors</i> flag (I/O logging disabled) is set. This setting is only supported by version 1.9.0 or higher.

GROUP PROVIDER PLUGINS

The **sudoers** plugin supports its own plugin interface to allow non-Unix group lookups which can query a group source other than the standard Unix group database. This can be used to implement support for the `nonunix_group` syntax described earlier.

Group provider plugins are specified via the *group_plugin* setting. The argument to *group_plugin* should consist of the plugin path, either fully-qualified or relative to the `/usr/libexec/sudo` directory, followed by any configuration options the plugin requires. These options (if specified) will be passed to the plugin's initialization function. If options are present, the string must be enclosed in double quotes ("").

The following group provider plugins are installed by default:

group_file

The *group_file* plugin supports an alternate group file that uses the same syntax as the `/etc/group` file. The path to the group file should be specified as an option to the plugin. For example, if the group file to be used is `/etc/sudo-group`:

```
Defaults group_plugin="group_file.so /etc/sudo-group"
```

system_group

The *system_group* plugin supports group lookups via the standard C library functions **getgrnam()** and **getgrid()**. This plugin can be used in instances where the user belongs to groups not present in the user's supplemental group vector. This plugin takes no options:

```
Defaults group_plugin=system_group.so
```

The group provider plugin API is described in detail in [sudo_plugin\(5\)](#).

LOG FORMAT

sudoers can log events in either JSON or *sudo* format, this section describes the *sudo* log format. Depending on *sudoers* configuration, **sudoers** can log events via [syslog\(3\)](#), to a local log file, or both. The log format is almost identical in both cases.

Accepted command log entries

Commands that *sudo* runs are logged using the following format (split into multiple lines for readability):

```
date hostname progname: username : TTY=ttyname ; PWD=cwd ; \
USER=runasuser ; GROUP=runasgroup ; TSID=loginid ; \
ENV=env_vars COMMAND=command
```

Where the fields are as follows:

date	The date the command was run. Typically, this is in the format “MMM, DD, HH:MM:SS”. If logging via syslog(3) , the actual date format is controlled by the syslog daemon. If logging to a file and the <i>log_year</i> option is enabled, the date will also include the year.
hostname	The name of the host sudo was run on. This field is only present when logging via syslog(3) .
progname	The name of the program, usually <i>sudo</i> or <i>sudoedit</i> . This field is only present when logging via syslog(3) .
username	The login name of the user who ran sudo .
ttyname	The short name of the terminal (e.g., “console”, “tty01”, or “pts/0”) sudo was run on, or “unknown” if there was no terminal present.
cwd	The current working directory that sudo was run in.
runasuser	The user the command was run as.
runasgroup	The group the command was run as if one was specified on the command line.
loginid	An I/O log identifier that can be used to replay the command's output. This is only present when the <i>log_input</i> or <i>log_output</i> option is enabled.
env_vars	A list of environment variables specified on the command line, if specified.
command	The actual command that was executed.

Messages are logged using the locale specified by *sudoers_locale*, which defaults to the “C” locale.

Denied command log entries

If the user is not allowed to run the command, the reason for the denial will follow the user name. Possible reasons include:

user NOT in sudoers

The user is not listed in the *sudoers* file.

user NOT authorized on host

The user is listed in the *sudoers* file but is not allowed to run commands on the host.

command not allowed

The user is listed in the *sudoers* file for the host but they are not allowed to run the specified command.

3 incorrect password attempts

The user failed to enter their password after 3 tries. The actual number of tries will vary based on the number of failed attempts and the value of the *passwd_tries* option.

a password is required

The **-n** option was specified but a password was required.

sorry, you are not allowed to set the following environment variables

The user specified environment variables on the command line that were not allowed by *sudoers*.

Error log entries

If an error occurs, **sudoers** will log a message and, in most cases, send a message to the administrator via email. Possible errors include:

parse error in /etc/sudoers near line N

sudoers encountered an error when parsing the specified file. In some cases, the actual error may be one line above or below the line number listed, depending on the type of error.

problem with defaults entries

The *sudoers* file contains one or more unknown Defaults settings. This does not prevent **sudo** from running, but the *sudoers* file should be checked using **visudo**.

timestamp owner (username): No such user

The time stamp directory owner, as specified by the *timestampowner* setting, could not be found in the password database.

unable to open/read /etc/sudoers

The *sudoers* file could not be opened for reading. This can happen when the *sudoers* file is located on a remote file system that maps user-ID 0 to a different value. Normally, **sudoers** tries to open the *sudoers* file using group permissions to avoid this problem. Consider either changing the ownership of /etc/sudoers or adding an argument like “*sudoers_uid=N*” (where ‘N’ is the user-ID that owns the *sudoers* file) to the end of the **sudoers** Plugin line in the *sudo.conf(5)* file.

unable to stat /etc/sudoers

The /etc/sudoers file is missing.

/etc/sudoers is not a regular file

The /etc/sudoers file exists but is not a regular file or symbolic link.

/etc/sudoers is owned by uid N, should be 0

The *sudoers* file has the wrong owner. If you wish to change the *sudoer* s file owner, please add “*sudoers_uid=N*” (where ‘N’ is the user-ID that owns the *sudoers* file) to the **sudoers** Plugin line in the *sudo.conf(5)* file.

/etc/sudoers is world writable

The permissions on the *sudoers* file allow all users to write to it. The *sudoers* file must not be world-writable, the default file mode is 0440 (readable by owner and group, writable by none). The default mode may be changed via the “*sudoers_mode*” option to the **sudoers** Plugin line in the *sudo.conf(5)* file.

/etc/sudoers is owned by gid N, should be 1

The **sudoers** file has the wrong group ownership. If you wish to change the **sudoers** file group ownership, please add “**sudoers_gid=N**” (where ‘N’ is the group-ID that owns the **sudoers** file) to the **sudoers** **Plugin** line in the **sudo.conf(5)** file.

unable to open /run/sudo/ts/username

sudoers was unable to read or create the user’s time stamp file. This can happen when *timestampowner* is set to a user other than root and the mode on **/run/ sudo** is not searchable by group or other. The default mode for **/run/ sudo** is 0711.

unable to write to /run/sudo/ts/username

sudoers was unable to write to the user’s time stamp file.

/run/ sudo/ ts is owned by uid X, should be Y

The time stamp directory is owned by a user other than *timestampowner*. This can occur when the value of *timestampowner* has been changed. **sudoers** will ignore the time stamp directory until the owner is corrected.

/run/ sudo/ ts is group writable

The time stamp directory is group-writable; it should be writable only by *timestampowner*. The default mode for the time stamp directory is 0700. **sudoers** will ignore the time stamp directory until the mode is corrected.

Notes on logging via syslog

By default, **sudoers** logs messages via **syslog(3)**. The *date*, *hostname*, and *programname* fields are added by the system’s **syslog()** function, not **sudoers** itself. As such, they may vary in format on different systems.

The maximum size of syslog messages varies from system to system. The *syslog_maxlen* setting can be used to change the maximum syslog message size from the default value of 980 bytes. For more information, see the description of *syslog_maxlen*.

Notes on logging to a file

If the *logfile* option is set, **sudoers** will log to a local file, such as **/var/ log/ sudo**. When logging to a file, **sudoers** uses a format similar to **syslog(3)**, with a few important differences:

1. The *programname* and *hostname* fields are not present.
2. If the *log_year* option is enabled, the date will also include the year.
3. Lines that are longer than *loglinelen* characters (80 by default) are word-wrapped and continued on the next line with a four character indent. This makes entries easier to read for a human being, but makes it more difficult to use **grep(1)** on the log files. If the *loglinelen* option is set to 0 (or negated with a ‘!’), word wrap will be disabled.

I/O LOG FILES

When I/O logging is enabled, **sudo** will run the command in a pseudo-terminal and log all user input and/or output, depending on which options are enabled. I/O can be logged either to the local machine or to a remote log server. For local logs, I/O is logged to the directory specified by the *iolog_dir* option (**/var/ log/ sudo- io** by default) using a unique session ID that is included in the **sudo** log line, prefixed with “*TSID=*”. The *iolog_file* option may be used to control the format of the session ID. For remote logs, the *log_servers* setting is used to specify one or more log servers running **sudo_logsrvd** or another server that implements the protocol described by **sudo_logsrv.proto(5)**.

For both local and remote I/O logs, each log is stored in a separate directory that contains the following files:

<code>log</code>	A text file containing information about the command. The first line consists of the following colon-delimited fields: the time the command was run, the name of the user who ran sudo , the name of the target user, the name of the target group (optional), the terminal that sudo was run from, and the number of lines and columns of the terminal. The second and third lines contain the working directory the command was run from and the path name of the command itself (with arguments if present).
<code>log.json</code>	A JSON-formatted file containing information about the command. This is similar to the <code>log</code> file but contains additional information and is easily extensible. The <code>log.json</code> file will be used by sudoreplay(8) in preference to the <code>log</code> file if it exists. The file may contain the following elements:
<code>timestamp</code>	A JSON object containing time the command was run. It consists of two values, <code>seconds</code> and <code>nanoseconds</code> .
<code>columns</code>	The number of columns of the terminal the command ran on, or zero if no terminal was present.
<code>command</code>	The fully-qualified path of the command that was run.
<code>lines</code>	The number of lines of the terminal the command ran on, or zero if no terminal was present.
<code>runargv</code>	A JSON array representing the command's argument vector as passed to the execve() system call.
<code>runenv</code>	A JSON array representing the command's environment as passed to the execve() system call.
<code>rungid</code>	The group ID the command ran as. This element is only present when the user specifies a group on the command line.
<code>rungroup</code>	The name of the group the command ran as. This element is only present when the user specifies a group on the command line.
<code>runuid</code>	The user ID the command ran as.
<code>runuser</code>	The name of the user the command ran as.
<code>submit cwd</code>	The current working directory at the time sudo was run.
<code>submit host</code>	The name of the host the command was run on.
<code>submit user</code>	The name of the user who ran the command via sudo .
<code>ttyname</code>	The path name of the terminal the user invoked sudo from. If the command was run in a pseudo-terminal, <code>ttyname</code> will be different from the terminal the command actually ran in.
<code>timing</code>	Timing information used to replay the session. Each line consists of the I/O log entry type and amount of time since the last entry, followed by type-specific data. The I/O log entry types and their corresponding type-specific data are:

0	standard input, number of bytes in the entry
1	standard output, number of bytes in the entry
2	standard error, number of bytes in the entry
3	terminal input, number of bytes in the entry
4	terminal output, number of bytes in the entry
5	window change, new number lines and columns
6	bug compatibility for sudo 1.8.7 terminal output
7	command suspend or resume, signal received
ttyin	Raw input from the user's terminal, exactly as it was received. No post-processing is performed. For manual viewing, you may wish to convert carriage return characters in the log to line feeds. For example: gunzip -c ttyin tr "\r" "\n"
stdin	The standard input when no terminal is present, or input redirected from a pipe or file.
ttyout	Output from the pseudo-terminal (what the command writes to the screen). Note that terminal-specific post-processing is performed before the data is logged. This means that, for example, line feeds are usually converted to line feed/carriage return pairs and tabs may be expanded to spaces.
stdout	The standard output when no terminal is present, or output redirected to a pipe or file.
stderr	The standard error redirected to a pipe or file.

All files other than **log** are compressed in gzip format unless the *compress_io* flag has been disabled. Due to buffering, it is not normally possible to display the I/O logs in real-time as the program is executing. The I/O log data will not be complete until the program run by **sudo** has exited or has been terminated by a signal. The *iolo g_flush* flag can be used to disable buffering, in which case I/O log data is written to disk as soon as it is available. The output portion of an I/O log file can be viewed with the **sudoreplay(8)** utility, which can also be used to list or search the available logs.

Note that user input may contain sensitive information such as passwords (even if they are not echoed to the screen), which will be stored in the log file unencrypted. In most cases, logging the command output via *log_output* or **LOG_OUTPUT** is all that is required.

Since each session's I/O logs are stored in a separate directory, traditional log rotation utilities cannot be used to limit the number of I/O logs. The simplest way to limit the number of I/O is by setting the *maxseq* option to the maximum number of logs you wish to store. Once the I/O log sequence number reaches *maxseq*, it will be reset to zero and **sudoers** will truncate and re-use any existing I/O logs.

FILES

/etc/sudo.conf	Sudo front-end configuration
/etc/sudoers	List of who can run what
/etc/group	Local groups file
/etc/netgroup	List of network groups
/var/log/sudo-io	I/O log files
/run/sudo/ts	Directory containing time stamps for the sudoers security policy
/var/lib/sudo/lectured	Directory containing lecture status files for the sudoers security policy
/etc/environment	Initial environment for -i mode on AIX and Linux systems

EXAMPLES

Below are example *sudoers* file entries. Admittedly, some of these are a bit contrived. First, we allow a few environment variables to pass and then define our *aliases*:

```
# Run X applications through sudo; HOME is used to find the
# .Xauthority file. Note that other programs use HOME to find
# configuration files and this may lead to privilege escalation!
Defaults env_keep += "DISPLAY HOME"

# User alias specification
User_Alias      FULLTIMERS = millert, mikef, dowdy
User_Alias      PARTTIMERS = bostley, jwfox, crawl
User_Alias      WEBADMIN = will, wendy, wim

# Runas alias specification
Runas_Alias    OP = root, operator
Runas_Alias    DB = oracle, sybase
Runas_Alias    ADMINGRP = adm, oper

# Host alias specification
Host_Alias     SPARC = bigtime, eclipse, moet, anchor :\
                SGI = grolsch, dandelion, black :\
                ALPHA = widget, thalamus, foobar :\
                HPPA = boa, nag, python
Host_Alias     CUNETS = 128.138.0.0/255.255.0.0
Host_Alias     CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias     SERVERS = primary, mail, www, ns
Host_Alias     CDROM = orion, perseus, hercules

# Cmnd alias specification
Cmnd_Alias    DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump, \
                /usr/sbin/restore, /usr/sbin/rrestore, \
                sha224:0GomF8mNN3wlDt1HD9XldjJ3SNgpFdbj01+NsQ== \
                /home/operator/bin/start_backups
Cmnd_Alias    KILL = /usr/bin/kill
Cmnd_Alias    PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias    SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias    HALT = /usr/sbin/halt
Cmnd_Alias    REBOOT = /usr/sbin/reboot
Cmnd_Alias    SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                /usr/local/bin/tcsh, /usr/bin/rsh, \
                /usr/local/bin/zsh
Cmnd_Alias    SU = /usr/bin/su
Cmnd_Alias    PAGERS = /usr/bin/more, /usr/bin/pg, /usr/bin/less
```

Here we override some of the compiled in default values. We want **sudo** to log via `syslog(3)` using the *auth* facility in all cases and for commands to be run with the target user's home directory as the working directory. We don't want to subject the full time staff to the **sudo** lecture and we want to allow them to run commands in a `chroot(2)` "sandbox" via the **-R** option. User **millert** need not provide a password and we don't want to reset the `LOGNAME` or `USER` environment variables when running commands as root. Additionally, on the machines in the *SERVERS* `Host_Alias`, we keep an additional local log file and make sure we log the year in each log line since the log entries will be kept around for several years. Lastly, we disable shell escapes for the commands in the *PAGERS* `Cmnd_Alias` (`/usr/bin/more`, `/usr/bin/pg` and

`/usr/bin/less`). Note that this will not effectively constrain users with **sudo ALL** privileges.

```
# Override built-in defaults
Defaults           syslog=auth,runcwd=~
Defaults>root      !set_logname
Defaults:FULLTIMERS !lecture,runchroot=*
Defaults:millert   !authenticate
Defaults@SERVERS   log_year, logfile=/var/log/sudo.log
Defaults!PAGERS    noexec
```

The *User specification* is the part that actually determines who may run what.

```
root          ALL = (ALL) ALL
%wheel        ALL = (ALL) ALL
```

We let **root** and any user in group **wheel** run any command on any host as any user.

```
FULLTIMERS     ALL = NOPASSWD: ALL
```

Full time sysadmins (**millert**, **mik ef**, and **dowdy**) may run any command on any host without authenticating themselves.

```
PARTTIMERS    ALL = ALL
```

Part time sysadmins (**bostley**, **jwfox**, and **crawl**) may run any command on any host but they must authenticate themselves first (since the entry lacks the NOPASSWD tag).

```
jack          CSNETS = ALL
```

The user **jack** may run any command on the machines in the *CSNETS* alias (the networks 128.138.243.0, 128.138.204.0, and 128.138.242.0). Of those networks, only 128.138.204.0 has an explicit netmask (in CIDR notation) indicating it is a class C network. For the other networks in *CSNETS*, the local machine's netmask will be used during matching.

```
lisa          CUNETS = ALL
```

The user **lisa** may run any command on any host in the *CUNETS* alias (the class B network 128.138.0.0).

```
operator       ALL = DUMPS, KILL, SHUTDOWN, HALT, REBOOT, PRINTING, \
                sudoedit /etc/printcap, /usr/oper/bin/
```

The **operator** user may run commands limited to simple maintenance. Here, those are commands related to backups, killing processes, the printing system, shutting down the system, and any commands in the directory `/usr/oper/bin/`. Note that one command in the `DUMPS` Cmnd_Alias includes a sha224 digest, `/home/operator/bin/start_backups`. This is because the directory containing the script is writable by the operator user. If the script is modified (resulting in a digest mismatch) it will no longer be possible to run it via **sudo**.

```
joe          ALL = /usr/bin/su operator
```

The user **joe** may only `su(1)` to operator.

```
pete          HPPA = /usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd *root*
%opers        ALL = (: ADMINGRP) /usr/sbin/
```

Users in the **opers** group may run commands in `/usr/sbin/` as themselves with any group in the `ADMINGRP` Runas_Alias (the **adm** and **oper** groups).

The user **pete** is allowed to change anyone's password except for root on the *HPPA* machines. Because command line arguments are matched as a single, concatenated string, the '*' wildcard will match *multiple* words. This example assumes that `passwd(1)` does not take multiple user names on the command line. Note that on GNU systems, options to `passwd(1)` may be specified after the user argument. As a result, this rule will also allow:

```
passwd username --expire
```

which may not be desirable.

```
bob           SPARC = (OP) ALL : SGI = (OP) ALL
```

The user **bob** may run anything on the *SPARC* and *SGI* machines as any user listed in the *OP Runas_Alias* (**root** and **operator**.)

```
jim           +biglab = ALL
```

The user **jim** may run any command on machines in the *biglab* netgroup. **sudo** knows that "biglab" is a netgroup due to the '+' prefix.

```
+secretaries  ALL = PRINTING, /usr/bin/adduser, /usr/bin/rmuser
```

Users in the **secretaries** netgroup need to help manage the printers as well as add and remove users, so they are allowed to run those commands on all machines.

```
fred          ALL = (DB) NOPASSWD: ALL
```

The user **fred** can run commands as any user in the *DB Runas_Alias* (**oracle** or **sybase**) without giving a password.

```
john          ALPHA = /usr/bin/su [!-]*, !/usr/bin/su *root*
```

On the *ALPHA* machines, user **john** may su to anyone except root but he is not allowed to specify any options to the `su(1)` command.

```
jen           ALL, !SERVERS = ALL
```

The user **jen** may run any command on any machine except for those in the *SERVERS Host_Alias* (primary, mail, www, and ns).

```
jill          SERVERS = /usr/bin/, !SU, !SHELLS
```

For any machine in the *SERVERS Host_Alias*, **jill** may run any commands in the directory `/usr/bin/` except for those commands belonging to the *SU* and *SHELLS Cmnd_Aliases*. While not specifically mentioned in the rule, the commands in the *PAGERS Cmnd_Alias* all reside in `/usr/bin` and have the *noexec* option set.

```
steve         CSNETS = (operator) /usr/local/op_commands/
```

The user **steve** may run any command in the directory `/usr/local/op_commands/` but only as user operator.

```
matt          valkyrie = KILL
```

On his personal workstation, **valkyrie**, **matt** needs to be able to kill hung processes.

```
WEBADMIN      www = (www) ALL, (root) /usr/bin/su www
```

On the host `www`, any user in the *WEBADMIN User_Alias* (will, wendy, and wim), may run any command as user `www` (which owns the web pages) or simply `su(1)` to `www`.

```
ALL          CDROM = NOPASSWD: /sbin/umount /CDROM,\n           /sbin/mount -o nosuid\,,nodev /dev/cd0a /CDROM
```

Any user may mount or unmount a CD-ROM on the machines in the CDROM Host_Alias (orion, perseus, hercules) without entering a password. This is a bit tedious for users to type, so it is a prime candidate for encapsulating in a shell script.

SECURITY NOTES

Limitations of the ‘!’ operator

It is generally not effective to “subtract” commands from **ALL** using the ‘!’ operator. A user can trivially circumvent this by copying the desired command to a different name and then executing that. For example:

```
bill    ALL = ALL, !SU, !SHELLS
```

Doesn’t really prevent **bill** from running the commands listed in **SU** or **SHELLS** since he can simply copy those commands to a different name, or use a shell escape from an editor or other program. Therefore, these kind of restrictions should be considered advisory at best (and reinforced by policy).

In general, if a user has sudo **ALL** there is nothing to prevent them from creating their own program that gives them a root shell (or making their own copy of a shell) regardless of any ‘!’ elements in the user specification.

Security implications of *fast_glob*

If the *fast_glob* option is in use, it is not possible to reliably negate commands where the path name includes globbing (aka wildcard) characters. This is because the C library’s `fnmatch(3)` function cannot resolve relative paths. While this is typically only an inconvenience for rules that grant privileges, it can result in a security issue for rules that subtract or revoke privileges.

For example, given the following *sudoers* file entry:

```
john    ALL = /usr/bin/passwd [a-zA-Z0-9]*, /usr/bin/chsh [a-zA-Z0-9]*, \
          /usr/bin/chfn [a-zA-Z0-9]*, !/usr/bin/* root
```

User **john** can still run `/usr/bin/passwd root` if *fast_glob* is enabled by changing to `/usr/bin` and running `./passwd root` instead.

Preventing shell escapes

Once **sudo** executes a program, that program is free to do whatever it pleases, including run other programs. This can be a security issue since it is not uncommon for a program to allow shell escapes, which lets a user bypass **sudo**’s access control and logging. Common programs that permit shell escapes include shells (obviously), editors, paginators, mail, and terminal programs.

There are four basic approaches to this problem:

restrict Avoid giving users access to commands that allow the user to run arbitrary commands. Many editors have a restricted mode where shell escapes are disabled, though **sudoedit** is a better solution to running editors via **sudo**. Due to the large number of programs that offer shell escapes, restricting users to the set of programs that do not is often unworkable.

intercept Many systems that support shared libraries have the ability to override default library functions by pointing an environment variable (usually `LD_PRELOAD`) to an alternate shared library. On such systems, **sudo**’s *intercept* functionality can be used to transparently intercept an attempt to run a new command, allow or deny it based on *sudoers* rules, and log the result. For example, this can be used to restrict the commands run from within a privileged shell. Note, however, that this applies only to dynamically-linked executables. Statically-linked executables and executables running under binary emulation are not affected. Also, most shells support built-in commands and the ability to read or write sensitive files that cannot be intercepted by **sudo**.

Currently, **sudo**'s *intercept* functionality only works for programs that use the **exec1()**, **execle()**, **execlp()**, **execv()**, **execve()**, **execvp()**, or **execvpe()** library functions to run the new command. This may be expanded in a future release of **sudo**. Because most dynamic loaders ignore **LD_PRELOAD** (or the equivalent) when running set-user-ID and set-group-ID programs, **sudoers** will not permit such programs to be run in *intercept* mode.

The *intercept* feature is known to work on Solaris, *BSD, Linux, macOS, HP-UX 11.x and AIX 5.3 and above. It should be supported on most operating systems that support the **LD_PRELOAD** environment variable. Check your operating system's manual pages for the dynamic linker (usually **ld.so**, **ld.so.1**, **dyld**, **dld.sl**, **rld**, or **loader**) to see if **LD_PRELOAD** is supported. It is *not* supported when **sudo**'s SELinux RBAC support is in use due to a fundamental incompatibility.

To enable intercept mode on a per-command basis, use the **INTERCEPT** tag as documented in the User Specification section above. Here is that example again:

```
chuck research = INTERCEPT: ALL
```

This allows user **chuck** to run any command on the machine "research" in intercept mode. Any commands run via shell escapes will be validated and logged by **sudo**. If you are unsure whether or not your system is capable of supporting *intercept*, you can always just try it out and check whether or not external commands run via a shell are logged when *intercept* is enabled.

log

There are two separate but related ways to log additional commands. The first is to enable I/O logging using the **log_output** flag. This will log the command's output but will not create an event log entry when the additional command is run. The second is to enable the **log_subcmds** flag in **sudoers** which will create an event log entry every time a new command is run. If I/O logging is also enabled, the log entry will include a time offset into the I/O log to indicate when the command was run. This offset can be passed to the **sudoreplay(8)** utility to replay the I/O log at the exact moment when the command was run. The **log_subcmds** flag uses the same mechanism as *intercept* (see above) and has the same limitations.

noexec

sudo's *noexec* functionality can be used to prevent a program run by **sudo** from executing any other programs. On most systems, it uses the same mechanism as *intercept* (see above) and thus the same caveats apply. The *noexec* functionality is capable of blocking execution of commands run via the **exec1()**, **execle()**, **execlp()**, **exec()**, **execv()**, **execve()**, **execveat()**, **execvP()**, **execvp()**, **execvpe()**, **fexecve()**, **popen()**, **posix_spawn()**, **posix_spawnp()**, **system()**, and **wordexp()** functions. On Linux, a **seccomp()** filter is used to implement *noexec*. On Solaris 10 and higher, *noexec* uses Solaris privileges instead of the **LD_PRELOAD** environment variable.

To enable *noexec* for a command, use the **NOEXEC** tag as documented in the User Specification section above. Here is that example again:

```
aaron shanty = NOEXEC: /usr/bin/more, /usr/bin/vi
```

This allows user **aaron** to run **/usr/bin/more** and **/usr/bin/vi** with *noexec* enabled. This will prevent those two commands from executing other commands (such as a shell). If you are unsure whether or not your system is capable of supporting *noexec* you can always just try it out and check whether shell escapes work when *noexec* is enabled.

Note that restricting shell escapes is not a panacea. Programs running as root are still capable of many potentially hazardous operations (such as changing or overwriting files) that could lead to unintended privilege escalation. In the specific case of an editor, a safer approach is to give the user permission to run **sudoedit** (see below).

Secure editing

The **sudoers** plugin includes **sudoedit** support which allows users to securely edit files with the editor of their choice. As **sudoedit** is a built-in command, it must be specified in the *sudoers* file without a leading path. However, it may take command line arguments just as a normal command does. Wildcards used in **sudoedit** command line arguments are expected to be path names, so a forward slash ('/') will not be matched by a wildcard.

Unlike other **sudo** commands, the editor is run with the permissions of the invoking user and with the environment unmodified. More information may be found in the description of the **-e** option in **sudo(8)**.

For example, to allow user operator to edit the “message of the day” file:

```
operator      sudoedit /etc/motd
```

The operator user then runs **sudoedit** as follows:

```
$ sudoedit /etc/motd
```

The editor will run as the operator user, not root, on a temporary copy of */etc/motd*. After the file has been edited, */etc/motd* will be updated with the contents of the temporary copy.

Users should *never* be granted **sudoedit** permission to edit a file that resides in a directory the user has write access to, either directly or via a wildcard. If the user has write access to the directory it is possible to replace the legitimate file with a link to another file, allowing the editing of arbitrary files. To prevent this, starting with version 1.8.16, symbolic links will not be followed in writable directories and **sudoedit** will refuse to edit a file located in a writable directory unless the *sudoedit_checkdir* option has been disabled or the invoking user is root. Additionally, in version 1.8.15 and higher, **sudoedit** will refuse to open a symbolic link unless either the *sudoedit_follow* option is enabled or the *sudoedit* command is prefixed with the FOLLOW tag in the *sudoers* file.

Time stamp file checks

sudoers will check the ownership of its time stamp directory (*/run/sudo/ts* by default) and ignore the directory’s contents if it is not owned by root or if it is writable by a user other than root. Older versions of **sudo** stored time stamp files in */tmp*; this is no longer recommended as it may be possible for a user to create the time stamp themselves on systems that allow unprivileged users to change the ownership of files they create.

While the time stamp directory *should* be cleared at reboot time, not all systems contain a */run* or */var/run* directory. To avoid potential problems, **sudoers** will ignore time stamp files that date from before the machine booted on systems where the boot time is available.

Some systems with graphical desktop environments allow unprivileged users to change the system clock. Since **sudoers** relies on the system clock for time stamp validation, it may be possible on such systems for a user to run **sudo** for longer than *timestamp_timeout* by setting the clock back. To combat this, **sudoers** uses a monotonic clock (which never moves backwards) for its time stamps if the system supports it.

sudoers will not honor time stamps set far in the future. Time stamps with a date greater than *current_time + 2 * TIMEOUT* will be ignored and **sudoers** will log and complain.

If the *timestamp_type* option is set to “tty”, the time stamp record includes the device number of the terminal the user authenticated with. This provides per-terminal granularity but time stamp records may still outlive the user’s session.

Unless the *timestamp_type* option is set to “global”, the time stamp record also includes the session ID of the process that last authenticated. This prevents processes in different terminal sessions from using the same time stamp record. On systems where a process’s start time can be queried, the start time of the session leader is recorded in the time stamp record. If no terminal is present or the *timestamp_type* option is set to “ppid”, the start time of the parent process is used instead. In most cases this will prevent a time stamp

record from being re-used without the user entering a password when logging out and back in again.

DEBUGGING

Versions 1.8.4 and higher of the **sudoers** plugin support a flexible debugging framework that can help track down what the plugin is doing internally if there is a problem. This can be configured in the **sudo.conf(5)** file.

The **sudoers** plugin uses the same debug flag format as the **sudo** front-end: *subsystem@priority*.

The priorities used by **sudoers**, in order of decreasing severity, are: *crit, err, warn, notice, diag, info, trace*, and *debug*. Each priority, when specified, also includes all priorities higher than it. For example, a priority of *notice* would include debug messages logged at *notice* and higher.

The following subsystems are used by the **sudoers** plugin:

<i>alias</i>	User_Alias, Runas_Alias, Host_Alias and Cmnd_Alias processing
<i>all</i>	matches every subsystem
<i>audit</i>	BSM and Linux audit code
<i>auth</i>	user authentication
<i>defaults</i>	<i>sudoers</i> file Defaults settings
<i>env</i>	environment handling
<i>ldap</i>	LDAP-based sudoers
<i>logging</i>	logging support
<i>match</i>	matching of users, groups, hosts, and netgroups in the <i>sudoers</i> file
<i>netif</i>	network interface handling
<i>nss</i>	network service switch handling in sudoers
<i>parser</i>	<i>sudoers</i> file parsing
<i>perms</i>	permission setting
<i>plugin</i>	The equivalent of <i>main</i> for the plugin.
<i>pty</i>	pseudo-terminal related code
<i>rbtree</i>	redblack tree internals
<i>sssd</i>	SSSD-based sudoers
<i>util</i>	utility functions

For example:

```
Debug sudoers.so /var/log/sudoers_debug match@info,nss@info
```

For more information, see the **sudo.conf(5)** manual.

SEE ALSO

ssh(1), su(1), fnmatch(3), glob(3), mktemp(3), strftime(3), sudo.conf(5), sudo_plugin(5), sudoers.ldap(5), sudoers_timestamp(5), sudo(8), visudo(8)

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

CAVEATS

The *sudoers* file should **always** be edited by the **visudo** utility which locks the file and checks for syntax errors. If *sudoer.s* contains syntax errors, **sudo** may refuse to run, which is a serious problem if **sudo** is your only method of obtaining superuser privileges. Recent versions of **sudoers** will attempt to recover after a syntax error by ignoring the rest of the line after encountering an error. Older versions of **sudo** will not run if *sudoers* contains a syntax error.

When using netgroups of machines (as opposed to users), if you store fully qualified host name in the netgroup (as is usually the case), you either need to have the machine's host name be fully qualified as returned by the `hostname` command or use the *fqdn* option in *sudoers*.

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudoers_timestamp — Sudoers Time Stamp Format

DESCRIPTION

The **sudoers** plugin uses per-user time stamp files for credential caching. Once a user has been authenticated, they may use **sudo** without a password for a short period of time (15 minutes unless overridden by the *timestamp_timeout* option). By default, **sudoers** uses a separate record for each terminal, which means that a user's login sessions are authenticated separately. The *timestamp_type* option can be used to select the type of time stamp record **sudoers** will use.

A multi-record time stamp file format was introduced in **sudo** 1.8.10 that uses a single file per user. Previously, a separate file was used for each user and terminal combination unless tty-based time stamps were disabled. The new format is extensible and records of multiple types and versions may coexist within the same file.

All records, regardless of type or version, begin with a 16-bit version number and a 16-bit record size.

Time stamp records have the following structure:

```
/* Time stamp entry types */
#define TS_GLOBAL          0x01 /* not restricted by tty or ppid */
#define TS_TTY              0x02 /* restricted by tty */
#define TS_PPID             0x03 /* restricted by ppid */
#define TS_LOCKEXCL         0x04 /* special lock record */

/* Time stamp flags */
#define TS_DISABLED          0x01 /* entry disabled */
#define TS_ANYUID            0x02 /* ignore uid, only valid in key */

struct timestamp_entry {
    unsigned short version;      /* version number */
    unsigned short size;         /* entry size */
    unsigned short type;         /* TS_GLOBAL, TS_TTY, TS_PPID */
    unsigned short flags;        /* TS_DISABLED, TS_ANYUID */
    uid_t auth_uid;             /* uid to authenticate as */
    pid_t sid;                  /* session ID associated with tty/ppid */
    struct timespec start_time; /* session/ppid start time */
    struct timespec ts;          /* time stamp (CLOCK_MONOTONIC) */
    union {
        dev_t ttydev;           /* tty device number */
        pid_t ppid;              /* parent pid */
    } u;
};
```

The `timestamp_entry` struct fields are as follows:

`version`

The version number of the `timestamp_entry` struct. New entries are created with a version number of 2. Records with different version numbers may coexist in the same file but are not inter-operable.

`size` The size of the record in bytes.

`type` The record type, currently `TS_GLOBAL`, `TS_TTY`, or `TS_PPID`.

flags Zero or more record flags which can be bit-wise ORed together. Supported flags are `TS_DISABLED`, for records disabled via `sudo -k` and `TS_ANYUID`, which is used only when matching records.

auth_uid

The user-ID that was used for authentication. Depending on the value of the `rootpw`, `runaspw` and `targetpw` options, the user-ID may be that of the invoking user, the root user, the default runas user or the target user.

sid The ID of the user's terminal session, if present. The session ID is only used when matching records of type `TS_TTY`.

start_time

The start time of the session leader for records of type `TS_TTY` or of the parent process for records of type `TS_PPID`. The `start_time` is used to help prevent re-use of a time stamp record after a user has logged out. Not all systems support a method to easily retrieve a process's start time. The `start_time` field was added in **sudoers** version 1.8.22 for the second revision of the `timestamp_entry` struct.

ts The actual time stamp. A monotonic time source (which does not move backward) is used if the system supports it. Where possible, **sudoers** uses a monotonic timer that increments even while the system is suspended. The value of `ts` is updated each time a command is run via `sudo`. If the difference between `ts` and the current time is less than the value of the `timestamp_timeout` option, no password is required.

u.ttydev

The device number of the terminal associated with the session for records of type `TS_TTY`.

u.ppid

The ID of the parent process for records of type `TS_PPID`.

LOCKING

In **sudoers** versions 1.8.10 through 1.8.14, the entire time stamp file was locked for exclusive access when reading or writing to the file. Starting in **sudoers** 1.8.15, individual records are locked in the time stamp file instead of the entire file and the lock is held for a longer period of time. This scheme is described below.

The first record in the time stamp file is of type `TS_LOCKEXCL` and is used as a `lock` record to prevent more than one `sudo` process from adding a new record at the same time. Once the desired time stamp record has been located or created (and locked), the `TS_LOCKEXCL` record is unlocked. The lock on the individual time stamp record, however, is held until authentication is complete. This allows **sudoers** to avoid prompting for a password multiple times when it is used more than once in a pipeline.

Records of type `TS_GLOBAL` cannot be locked for a long period of time since doing so would interfere with other `sudo` processes. Instead, a separate lock record is used to prevent multiple `sudo` processes using the same terminal (or parent process ID) from prompting for a password at the same time.

SEE ALSO

`sudoers(5)`, `sudo(8)`

HISTORY

Originally, `sudo` used a single zero-length file per user and the file's modification time was used as the time stamp. Later versions of `sudo` added restrictions on the ownership of the time stamp files and directory as well as checks on the validity of the time stamp itself. Notable changes were introduced in the following `sudo` versions:

1.4.0 Support for tty-based time stamp file was added by appending the terminal name to the time stamp file name.

1.6.2 The time stamp file was replaced by a per-user directory which contained any tty-based time stamp files.

1.6.3p2

The target user name was added to the time stamp file name when the *targetpw* option was set.

1.7.3 Information about the terminal device was stored in tty-based time stamp files for validity checks. This included the terminal device numbers, inode number and, on systems where it was not updated when the device was written to, the inode change time. This helped prevent re-use of the time stamp file after logout.

1.8.6p7

The terminal session ID was added to tty-based time stamp files to prevent re-use of the time stamp by the same user in a different terminal session. It also helped prevent re-use of the time stamp file on systems where the terminal device's inode change time was updated by writing.

1.8.10

A new, multi-record time stamp file format was introduced that uses a single file per user. The terminal device's change time was not included since most systems now update the change time after a write is performed as required by POSIX.

1.8.15

Individual records are locked in the time stamp file instead of the entire file and the lock is held until authentication is complete.

1.8.22

The start time of the terminal session leader or parent process is now stored in non-global time stamp records. This prevents re-use of the time stamp file after logout in most cases.

Support was added for the kernel-based tty time stamps available in OpenBSD which do not use an on-disk time stamp file.

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudo is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

sudoreplay — replay sudo session logs

SYNOPSIS

```
sudoreplay [ -FhnRS ] [-d dir] [-f filter] [-m num] [-s num] ID[@offset]
sudoreplay [ -h ] [-d dir] -l [search expression]
```

DESCRIPTION

sudoreplay plays back or lists the output logs created by **sudo**. When replaying, **sudoreplay** can play the session back in real-time, or the playback speed may be adjusted (faster or slower) based on the command line options.

The *ID* should either be a six character sequence of digits and upper case letters, e.g., 0100A5 or a path name. The *ID* may include an optional @*offset* suffix which may be used to start replaying at a specific time offset. The @*offset* is specified as a number in seconds since the start of the session with an optional decimal fraction.

Path names may be relative to the I/O log directory /var/log/sudo-io (unless overridden by the **-d** option) or fully qualified, beginning with a ‘/’ character. When a command is run via **sudo** with *log_output* enabled in the *sudoers* file, a TSID=ID string is logged via syslog or to the **sudo** log file. The *ID* may also be determined using **sudoreplay**'s list mode.

In list mode, **sudoreplay** can be used to find the ID of a session based on a number of criteria such as the user, tty, or command run.

In replay mode, if the standard input and output are connected to a terminal and the **-n** option is not specified, **sudoreplay** will operate interactively. In interactive mode, **sudoreplay** will attempt to adjust the terminal size to match that of the session and write directly to the terminal (not all terminals support this). Additionally, it will poll the keyboard and act on the following keys:

- ‘\n’ or ‘\r’ Skip to the next replay event; useful for long pauses.
- ‘ ’ (space) Pause output; press any key to resume.
- ‘<’ Reduce the playback speed by one half.
- ‘>’ Double the playback speed.

The session can be interrupted via control-C. When the session has finished, the terminal is restored to its original size if it was changed during playback.

The options are as follows:

- d dir, --directory=dir**
Store session logs in *dir* instead of the default, /var/log/sudo-io.
- f filter, --filter=filter**
Select which I/O type(s) to display. By default, **sudoreplay** will display the command's standard output, standard error, and tty output. The *filter* argument is a comma-separated list, consisting of one or more of following: *stdin*, *stdout*, *stderr*, *ttyin*, and *ttyout*.
- F, --follow**
Enable “follow mode”. When replaying a session, **sudoreplay** will ignore end-of-file and keep replaying until the log is complete. This can be used to replay a session that is still in progress, similar to “tail -f”. An I/O log file is considered to be complete when the write bits have been cleared on the session's timing file. Note that versions of **sudo** prior to 1.9.1 do not clear the write bits upon completion.

-h, --help

Display a short help message to the standard output and exit.

-l, --list [search expression]

Enable “list mode”. In this mode, **sudoreplay** will list available sessions in a format similar to the **sudo** log file format, sorted by file name (or sequence number). If a *search expression* is specified, it will be used to restrict the IDs that are displayed. An expression is composed of the following predicates:

command pattern

Evaluates to true if the command run matches the POSIX extended regular expression *pattern*.

cwd directory

Evaluates to true if the command was run with the specified current working directory.

fromdate date

Evaluates to true if the command was run on or after *date*. See**Date and time format** for a description of supported date and time formats.

group runas_group

Evaluates to true if the command was run with the specified *runas_group*. Note that unless a *runas_group* was explicitly specified when **sudo** was run this field will be empty in the log.

host hostname

Evaluates to true if the command was run on the specified *hostname*.

runas runas_user

Evaluates to true if the command was run as the specified *runas_user*. Note that **sudo** runs commands as user *root* by default.

todate date

Evaluates to true if the command was run on or prior to *date*. See**Date and time format** for a description of supported date and time formats.

tty tty name

Evaluates to true if the command was run on the specified terminal device. The *tty* *name* should be specified without the */dev/* prefix, e.g., *tty01* instead of */dev/tty01*.

user user name

Evaluates to true if the ID matches a command run by *user name*.

Predicates may be abbreviated to the shortest unique string.

Predicates may be combined using *and*, *or*, and *!* operators as well as ‘(’ and ‘)’ grouping (note that parentheses must generally be escaped from the shell). The *and* operator is optional, adjacent predicates have an implied *and* unless separated by an *or*.

-m, --max-wait max_wait

Specify an upper bound on how long to wait between key presses or output data. By default, **sudoreplay** will accurately reproduce the delays between key presses or program output. However, this can be tedious when the session includes long pauses. When the **-m** option is specified, **sudoreplay** will limit these pauses to at most *max_wait* seconds. The value may be specified as a floating point number, e.g., 2.5. A *max_wait* of zero or less will eliminate the pauses entirely.

-n, --non-interactive

Do not prompt for user input or attempt to re-size the terminal. The session is written to the standard output, not directly to the user's terminal.

-R, --no-resize

Do not attempt to re-size the terminal to match the terminal size of the session.

-S, --suspend-wait

Wait while the command was suspended. By default, **sudoreplay** will ignore the time interval between when the command was suspended and when it was resumed. If the **-S** option is specified, **sudoreplay** will wait instead.

-s, --speed speed_factor

This option causes **sudoreplay** to adjust the number of seconds it will wait between key presses or program output. This can be used to slow down or speed up the display. For example, a *speed_factor* of 2 would make the output twice as fast whereas a *speed_factor* of .5 would make the output twice as slow.

-v, --version

Print the **sudoreplay** versions version number and exit.

Date and time format

The time and date may be specified multiple ways, common formats include:

HH:MM:SS am MM/DD/CCYY timezone

24 hour time may be used in place of am/pm.

HH:MM:SS am Month, Day Year timezone

24 hour time may be used in place of am/pm, and month and day names may be abbreviated. Note that month and day of the week names must be specified in English.

CCYY-MM-DD HH:MM:SS

ISO time format

DD Month CCYY HH:MM:SS

The month name may be abbreviated.

Either time or date may be omitted, the am/pm and timezone are optional. If no date is specified, the current day is assumed; if no time is specified, the first second of the specified date is used. The less significant parts of both time and date may also be omitted, in which case zero is assumed.

The following are all valid time and date specifications:

now The current time and date.

tomorrow

Exactly one day from now.

yesterday

24 hours ago.

2 hours ago

2 hours ago.

next Friday

The first second of the Friday in the next (upcoming) week. Not to be confused with "this Friday" which would match the Friday of the current week.

last week

The current time but 7 days ago. This is equivalent to “a week ago”.

a fortnight ago

The current time but 14 days ago.

10:01 am 9/17/2009

10:01 am, September 17, 2009.

10:01 am

10:01 am on the current day.

10 10:00 am on the current day.

9/17/2009

00:00 am, September 17, 2009.

10:01 am Sep 17, 2009

10:01 am, September 17, 2009.

Note that relative time specifications do not always work as expected. For example, the “next” qualifier is intended to be used in conjunction with a day such as “next Monday”. When used with units of weeks, months, years, etc the result will be one more than expected. For example, “next week” will result in a time exactly two weeks from now, which is probably not what was intended. This will be addressed in a future version of **sudoreplay**.

Debugging sudoreplay

sudoreplay versions 1.8.4 and higher support a flexible debugging framework that is configured via Debug lines in the **sudo.conf(5)** file.

For more information on configuring **sudo.conf(5)**, please refer to its manual.

FILES

/etc/sudo.conf	Debugging framework configuration
/var/log/sudo-io	The default I/O log directory.
/var/log/sudo-io/00/00/01/log	Example session log info.
/var/log/sudo-io/00/00/01/log.json	Example session log info (JSON format).
/var/log/sudo-io/00/00/01/stdin	Example session standard input log.
/var/log/sudo-io/00/00/01/stdout	Example session standard output log.
/var/log/sudo-io/00/00/01/stderr	Example session standard error log.
/var/log/sudo-io/00/00/01/ttyin	Example session tty input file.
/var/log/sudo-io/00/00/01/ttyout	Example session tty output file.

```
/var/log/sudo-io/00/00/01/timing  
Example session timing file.
```

Note that the *stdin*, *stdout* and *stderr* files will be empty unless **sudo** was used as part of a pipeline for a particular command.

EXAMPLES

List sessions run by user *millert*:

```
# sudoreplay -l user millert
```

List sessions run by user *bob* with a command containing the string vi:

```
# sudoreplay -l user bob command vi
```

List sessions run by user *jeff* that match a regular expression:

```
# sudoreplay -l user jeff command '/bin/[a-z]*sh'
```

List sessions run by *jeff* or *bob* on the console:

```
# sudoreplay -l ( user jeff or user bob ) tty console
```

SEE ALSO

[script\(1\)](#), [sudo.conf\(5\)](#), [sudo\(8\)](#)

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (<https://www.sudo.ws/contributors.html>) for an exhaustive list of people who have contributed to **sudo**.

BUGS

If you feel you have found a bug in **sudoreplay**, please submit a bug report at <https://bugzilla.sudo.ws/>

SUPPORT

Limited free support is available via the sudo-users mailing list, see <https://www.sudo.ws/mailman/listinfo/sudo-users> to subscribe or search the archives.

DISCLAIMER

sudoreplay is provided “AS IS” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or <https://www.sudo.ws/license.html> for complete details.

NAME

swapon, swapoff – start/stop swapping to file/device

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/swap.h>
int swapon(const char *path, int swapflags);
int swapoff(const char *path);
```

DESCRIPTION

swapon() sets the swap area to the file or block device specified by *path*. **swapoff()** stops swapping to the file or block device specified by *path*.

If the **SWAP_FLAG_PREFER** flag is specified in the **swapon()** *swapflags* argument, the new swap area will have a higher priority than default. The priority is encoded within *swapflags* as:

```
(prio << SWAP_FLAG_PRIO_SHIFT) & SWAP_FLAG_PRIO_MASK
```

If the **SWAP_FLAG_DISCARD** flag is specified in the **swapon()** *swapflags* argument, freed swap pages will be discarded before they are reused, if the swap device supports the discard or trim operation. (This may improve performance on some Solid State Devices, but often it does not.) See also NOTES.

These functions may be used only by a privileged process (one having the **CAP_SYS_ADMIN** capability).

Priority

Each swap area has a priority, either high or low. The default priority is low. Within the low-priority areas, newer areas are even lower priority than older areas.

All priorities set with *swapflags* are high-priority, higher than default. They may have any nonnegative value chosen by the caller. Higher numbers mean higher priority.

Swap pages are allocated from areas in priority order, highest priority first. For areas with different priorities, a higher-priority area is exhausted before using a lower-priority area. If two or more areas have the same priority, and it is the highest priority available, pages are allocated on a round-robin basis between them.

As of Linux 1.3.6, the kernel usually follows these rules, but there are exceptions.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EBUSY

(for **swapon()**) The specified *path* is already being used as a swap area.

EINVAL

The file *path* exists, but refers neither to a regular file nor to a block device;

EINVAL

(**swapon()**) The indicated path does not contain a valid swap signature or resides on an in-memory filesystem such as **tmpfs**(5).

EINVAL (since Linux 3.4)

(**swapon()**) An invalid flag value was specified in *swapflags*.

EINVAL

(**swapoff()**) *path* is not currently a swap area.

ENFILE

The system-wide limit on the total number of open files has been reached.

ENOENT

The file *path* does not exist.

ENOMEM

The system has insufficient memory to start swapping.

EPERM

The caller does not have the **CAP_SYS_ADMIN** capability. Alternatively, the maximum number of swap files are already in use; see NOTES below.

STANDARDS

These functions are Linux-specific and should not be used in programs intended to be portable. The second *swapflags* argument was introduced in Linux 1.3.2.

NOTES

The partition or path must be prepared with **mkswap(8)**.

There is an upper limit on the number of swap files that may be used, defined by the kernel constant **MAX_SWAPFILES**. Before Linux 2.4.10, **MAX_SWAPFILES** has the value 8; since Linux 2.4.10, it has the value 32. Since Linux 2.6.18, the limit is decreased by 2 (thus: 30) if the kernel is built with the **CONFIG_MIGRATION** option (which reserves two swap table entries for the page migration features of **mbind(2)** and **migrate_pages(2)**). Since Linux 2.6.32, the limit is further decreased by 1 if the kernel is built with the **CONFIG_MEMORY_FAILURE** option. Since Linux 5.14, the limit is further decreased by 4 if the kernel is built with the **CONFIG_DEVICE_PRIVATE** option.

Discard of swap pages was introduced in Linux 2.6.29, then made conditional on the **SWAP_FLAG_DISCARD** flag in Linux 2.6.36, which still discards the entire swap area when **swapon()** is called, even if that flag bit is not set.

SEE ALSO

mkswap(8), **swapoff(8)**, **swapon(8)**

NAME

swapon, **swapoff** – enable/disable devices and files for paging and swapping

SYNOPSIS

swapon [options] [*specialfile...*]

swapoff [**-va**] [*specialfile...*]

DESCRIPTION

swapon is used to specify devices on which paging and swapping are to take place.

The device or file used is given by the *specialfile* parameter. It may be of the form **-L** *label* or **-U** *uuid* to indicate a device by label or uuid.

Calls to **swapon** normally occur in the system boot scripts making all swap devices available, so that the paging and swapping activity is interleaved across several devices and files.

swapoff disables swapping on the specified devices and files. When the **-a** flag is given, swapping is disabled on all known swap devices and files (as found in */proc/swaps* or */etc/fstab*).

OPTIONS**-a, --all**

All devices marked as "swap" in */etc/fstab* are made available, except for those with the "noauto" option. Devices that are already being used as swap are silently skipped.

-d, --discard[=policy]

Enable swap discards, if the swap backing device supports the discard or trim operation. This may improve performance on some Solid State Devices, but often it does not. The option allows one to select between two available swap discard policies:

--discard=once

to perform a single-time discard operation for the whole swap area at swapon; or

--discard=pages

to asynchronously discard freed swap pages before they are available for reuse.

If no policy is selected, the default behavior is to enable both discard types. The */etc/fstab* mount options **discard**, **discard=once**, or **discard=pages** may also be used to enable discard flags.

-e, --ifexists

Silently skip devices that do not exist. The */etc/fstab* mount option **nofail** may also be used to skip non-existing device.

-f, --fixpgsz

Reinitialize (exec **mkswap**) the swap space if its page size does not match that of the current running kernel. **mkswap(8)** initializes the whole device and does not check for bad blocks.

-L *label*

Use the partition that has the specified *label*. (For this, access to */proc/partitions* is needed.)

-o, --options *opts*

Specify swap options by an *fstab*-compatible comma-separated string. For example:

swapon -o pri=1,discard=pages,nofail /dev/sda2

The *opts* string is evaluated last and overrides all other command line options.

-p, --priority *priority*

Specify the priority of the swap device. *priority* is a value between -1 and 32767. Higher numbers indicate higher priority. See **swapon(2)** for a full description of swap priorities. Add **pri=value** to the option field of */etc/fstab* for use with **swapon -a**. When no priority is defined, it defaults to -1.

-s, --summary

Display swap usage summary by device. Equivalent to **cat /proc/swaps**. This output format is DEPRECATED in favour of **--show** that provides better control on output data.

--show[=column...]

Display a definable table of swap areas. See the **--help** output for a list of available columns.

--output-all

Output all available columns.

--noheadings

Do not print headings when displaying **--show** output.

--raw

Display **--show** output without aligning table columns.

--bytes

Display swap size in bytes in **--show** output instead of in user-friendly units.

-U *uuid*

Use the partition that has the specified *uuid*.

-v, --verbose

Be verbose.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

EXIT STATUS

swapoff has the following exit status values since v2.36:

0

success

2

system has insufficient memory to stop swapping (OOM)

4

swapoff(2) syscall failed for another reason

8

non-**swapoff(2)** syscall system error (out of memory, ...)

16

usage or syntax error

32

all swapoff failed on **--all**

64

some swapoff succeeded on **--all**

The command **swapoff --all** returns 0 (all succeeded), 32 (all failed), or 64 (some failed, some succeeded).

+ The old versions before v2.36 has no documented exit status, 0 means success in all versions.

ENVIRONMENT

LIBMOUNT_DEBUG=all

enables **libmount** debug output.

LIBBLKID_DEBUG=all

enables **libblkid** debug output.

FILES

/dev/sd??

standard paging devices

/etc/fstab

ascii filesystem description table

NOTES

Files with holes

The swap file implementation in the kernel expects to be able to write to the file directly, without the assistance of the filesystem. This is a problem on files with holes or on copy-on-write files on filesystems like Btrfs.

Commands like **cp(1)** or **truncate(1)** create files with holes. These files will be rejected by **swapon**.

Preallocated files created by **fallocate(1)** may be interpreted as files with holes too depending of the filesystem. Preallocated swap files are supported on XFS since Linux 4.18.

The most portable solution to create a swap file is to use **dd(1)** and */dev/zero*.

Btrfs

Swap files on Btrfs are supported since Linux 5.0 on files with **nocow** attribute. See the **btrfs(5)** manual page for more details.

NFS

Swap over **NFS** may not work.

Suspend

swapon automatically detects and rewrites a swap space signature with old software suspend data (e.g., **S1SUSPEND**, **S2SUSPEND**, ...). The problem is that if we don't do it, then we get data corruption the next time an attempt at unsuspending is made.

HISTORY

The **swapon** command appeared in 4.0BSD.

SEE ALSO

swapoff(2), **swapon(2)**, **fstab(5)**, **init(8)**, **fallocate(1)**, **mkswap(8)**, **mount(8)**, **rc(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **swapon** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](#) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

swapon, swapoff – start/stop swapping to file/device

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/swap.h>
int swapon(const char *path, int swapflags);
int swapoff(const char *path);
```

DESCRIPTION

swapon() sets the swap area to the file or block device specified by *path*. **swapoff()** stops swapping to the file or block device specified by *path*.

If the **SWAP_FLAG_PREFER** flag is specified in the **swapon()** *swapflags* argument, the new swap area will have a higher priority than default. The priority is encoded within *swapflags* as:

```
(prio << SWAP_FLAG_PRIO_SHIFT) & SWAP_FLAG_PRIO_MASK
```

If the **SWAP_FLAG_DISCARD** flag is specified in the **swapon()** *swapflags* argument, freed swap pages will be discarded before they are reused, if the swap device supports the discard or trim operation. (This may improve performance on some Solid State Devices, but often it does not.) See also NOTES.

These functions may be used only by a privileged process (one having the **CAP_SYS_ADMIN** capability).

Priority

Each swap area has a priority, either high or low. The default priority is low. Within the low-priority areas, newer areas are even lower priority than older areas.

All priorities set with *swapflags* are high-priority, higher than default. They may have any nonnegative value chosen by the caller. Higher numbers mean higher priority.

Swap pages are allocated from areas in priority order, highest priority first. For areas with different priorities, a higher-priority area is exhausted before using a lower-priority area. If two or more areas have the same priority, and it is the highest priority available, pages are allocated on a round-robin basis between them.

As of Linux 1.3.6, the kernel usually follows these rules, but there are exceptions.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

EBUSY

(for **swapon()**) The specified *path* is already being used as a swap area.

EINVAL

The file *path* exists, but refers neither to a regular file nor to a block device;

EINVAL

(**swapon()**) The indicated path does not contain a valid swap signature or resides on an in-memory filesystem such as **tmpfs**(5).

EINVAL (since Linux 3.4)

(**swapon()**) An invalid flag value was specified in *swapflags*.

EINVAL

(**swapoff()**) *path* is not currently a swap area.

ENFILE

The system-wide limit on the total number of open files has been reached.

ENOENT

The file *path* does not exist.

ENOMEM

The system has insufficient memory to start swapping.

EPERM

The caller does not have the **CAP_SYS_ADMIN** capability. Alternatively, the maximum number of swap files are already in use; see NOTES below.

STANDARDS

These functions are Linux-specific and should not be used in programs intended to be portable. The second *swapflags* argument was introduced in Linux 1.3.2.

NOTES

The partition or path must be prepared with **mkswap(8)**.

There is an upper limit on the number of swap files that may be used, defined by the kernel constant **MAX_SWAPFILES**. Before Linux 2.4.10, **MAX_SWAPFILES** has the value 8; since Linux 2.4.10, it has the value 32. Since Linux 2.6.18, the limit is decreased by 2 (thus: 30) if the kernel is built with the **CONFIG_MIGRATION** option (which reserves two swap table entries for the page migration features of **mbind(2)** and **migrate_pages(2)**). Since Linux 2.6.32, the limit is further decreased by 1 if the kernel is built with the **CONFIG_MEMORY_FAILURE** option. Since Linux 5.14, the limit is further decreased by 4 if the kernel is built with the **CONFIG_DEVICE_PRIVATE** option.

Discard of swap pages was introduced in Linux 2.6.29, then made conditional on the **SWAP_FLAG_DISCARD** flag in Linux 2.6.36, which still discards the entire swap area when **swapon()** is called, even if that flag bit is not set.

SEE ALSO

mkswap(8), **swapoff(8)**, **swapon(8)**

NAME

swapon, **swapoff** – enable/disable devices and files for paging and swapping

SYNOPSIS

swapon [options] [*specialfile...*]

swapoff [-va] [*specialfile...*]

DESCRIPTION

swapon is used to specify devices on which paging and swapping are to take place.

The device or file used is given by the *specialfile* parameter. It may be of the form **-L** *label* or **-U** *uuid* to indicate a device by label or uuid.

Calls to **swapon** normally occur in the system boot scripts making all swap devices available, so that the paging and swapping activity is interleaved across several devices and files.

swapoff disables swapping on the specified devices and files. When the **-a** flag is given, swapping is disabled on all known swap devices and files (as found in */proc/swaps* or */etc/fstab*).

OPTIONS**-a, --all**

All devices marked as "swap" in */etc/fstab* are made available, except for those with the "noauto" option. Devices that are already being used as swap are silently skipped.

-d, --discard[=policy]

Enable swap discards, if the swap backing device supports the discard or trim operation. This may improve performance on some Solid State Devices, but often it does not. The option allows one to select between two available swap discard policies:

--discard=once

to perform a single-time discard operation for the whole swap area at swapon; or

--discard=pages

to asynchronously discard freed swap pages before they are available for reuse.

If no policy is selected, the default behavior is to enable both discard types. The */etc/fstab* mount options **discard**, **discard=once**, or **discard=pages** may also be used to enable discard flags.

-e, --ifexists

Silently skip devices that do not exist. The */etc/fstab* mount option **nofail** may also be used to skip non-existing device.

-f, --fixpgsz

Reinitialize (exec **mkswap**) the swap space if its page size does not match that of the current running kernel. **mkswap(8)** initializes the whole device and does not check for bad blocks.

-L *label*

Use the partition that has the specified *label*. (For this, access to */proc/partitions* is needed.)

-o, --options *opts*

Specify swap options by an *fstab*-compatible comma-separated string. For example:

swapon -o pri=1,discard=pages,nofail /dev/sda2

The *opts* string is evaluated last and overrides all other command line options.

-p, --priority *priority*

Specify the priority of the swap device. *priority* is a value between -1 and 32767. Higher numbers indicate higher priority. See **swapon(2)** for a full description of swap priorities. Add **pri=value** to the option field of */etc/fstab* for use with **swapon -a**. When no priority is defined, it defaults to -1.

-s, --summary

Display swap usage summary by device. Equivalent to **cat /proc/swaps**. This output format is DEPRECATED in favour of **--show** that provides better control on output data.

--show[=column...]

Display a definable table of swap areas. See the **--help** output for a list of available columns.

--output-all

Output all available columns.

--noheadings

Do not print headings when displaying **--show** output.

--raw

Display **--show** output without aligning table columns.

--bytes

Display swap size in bytes in **--show** output instead of in user-friendly units.

-U *uuid*

Use the partition that has the specified *uuid*.

-v, --verbose

Be verbose.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

EXIT STATUS

swapoff has the following exit status values since v2.36:

0

success

2

system has insufficient memory to stop swapping (OOM)

4

swapoff(2) syscall failed for another reason

8

non-**swapoff(2)** syscall system error (out of memory, ...)

16

usage or syntax error

32

all swapoff failed on **--all**

64

some swapoff succeeded on **--all**

The command **swapoff --all** returns 0 (all succeeded), 32 (all failed), or 64 (some failed, some succeeded).

+ The old versions before v2.36 has no documented exit status, 0 means success in all versions.

ENVIRONMENT

LIBMOUNT_DEBUG=all

enables **libmount** debug output.

LIBBLKID_DEBUG=all

enables **libblkid** debug output.

FILES

/dev/sd??

standard paging devices

/etc/fstab

ascii filesystem description table

NOTES

Files with holes

The swap file implementation in the kernel expects to be able to write to the file directly, without the assistance of the filesystem. This is a problem on files with holes or on copy-on-write files on filesystems like Btrfs.

Commands like **cp(1)** or **truncate(1)** create files with holes. These files will be rejected by **swapon**.

Preallocated files created by **fallocate(1)** may be interpreted as files with holes too depending of the filesystem. Preallocated swap files are supported on XFS since Linux 4.18.

The most portable solution to create a swap file is to use **dd(1)** and */dev/zero*.

Btrfs

Swap files on Btrfs are supported since Linux 5.0 on files with **nocow** attribute. See the **btrfs(5)** manual page for more details.

NFS

Swap over **NFS** may not work.

Suspend

swapon automatically detects and rewrites a swap space signature with old software suspend data (e.g., **S1SUSPEND**, **S2SUSPEND**, ...). The problem is that if we don't do it, then we get data corruption the next time an attempt at unsuspending is made.

HISTORY

The **swapon** command appeared in 4.0BSD.

SEE ALSO

swapoff(2), **swapon(2)**, **fstab(5)**, **init(8)**, **fallocate(1)**, **mkswap(8)**, **mount(8)**, **rc(8)**

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **swapon** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](#) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

sysinfo – return system information

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/sysinfo.h>
int sysinfo(struct sysinfo *info);
```

DESCRIPTION

sysinfo() returns certain statistics on memory and swap usage, as well as the load average.

Until Linux 2.3.16, **sysinfo()** returned information in the following structure:

```
struct sysinfo {
    long uptime;           /* Seconds since boot */
    unsigned long loads[3]; /* 1, 5, and 15 minute load averages */
    unsigned long totalram; /* Total usable main memory size */
    unsigned long freeram; /* Available memory size */
    unsigned long sharedram; /* Amount of shared memory */
    unsigned long bufferram; /* Memory used by buffers */
    unsigned long totalswap; /* Total swap space size */
    unsigned long freeswap; /* Swap space still available */
    unsigned short procs; /* Number of current processes */
    char _f[22];           /* Pads structure to 64 bytes */
};
```

In the above structure, the sizes of the memory and swap fields are given in bytes.

Since Linux 2.3.23 (i386) and Linux 2.3.48 (all architectures) the structure is:

```
struct sysinfo {
    long uptime;           /* Seconds since boot */
    unsigned long loads[3]; /* 1, 5, and 15 minute load averages */
    unsigned long totalram; /* Total usable main memory size */
    unsigned long freeram; /* Available memory size */
    unsigned long sharedram; /* Amount of shared memory */
    unsigned long bufferram; /* Memory used by buffers */
    unsigned long totalswap; /* Total swap space size */
    unsigned long freeswap; /* Swap space still available */
    unsigned short procs; /* Number of current processes */
    unsigned long totalhigh; /* Total high memory size */
    unsigned long freehigh; /* Available high memory size */
    unsigned int mem_unit; /* Memory unit size in bytes */
    char _f[20-2*sizeof(long)-sizeof(int)]; /* Padding to 64 bytes */
};
```

In the above structure, sizes of the memory and swap fields are given as multiples of *mem_unit* bytes.

RETURN VALUE

On success, **sysinfo()** returns zero. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS**EFAULT**

info is not a valid address.

VERSIONS

sysinfo() first appeared in Linux 0.98.pl6.

STANDARDS

This function is Linux-specific, and should not be used in programs intended to be portable.

NOTES

All of the information provided by this system call is also available via */proc/meminfo* and */proc/loadavg*.

SEE ALSO

proc(5)

NAME

systemctl – Control the systemd system and service manager

SYNOPSIS

systemctl [OPTIONS...] COMMAND [UNIT...]

DESCRIPTION

systemctl may be used to introspect and control the state of the "systemd" system and service manager.

Please refer to **systemd(1)** for an introduction into the basic concepts and functionality this tool manages.

COMMANDS

The following commands are understood:

Unit Commands (Introspection and Modification)

list-units [*PATTERN...*]

List units that **systemd** currently has in memory. This includes units that are either referenced directly or through a dependency, units that are pinned by applications programmatically, or units that were active in the past and have failed. By default only units which are active, have pending jobs, or have failed are shown; this can be changed with option **--all**. If one or more *PATTERNs* are specified, only units matching one of them are shown. The units that are shown are additionally filtered by **--type=** and **--state=** if those options are specified.

Note that this command does not show unit templates, but only instances of unit templates. Units templates that aren't instantiated are not runnable, and will thus never show up in the output of this command. Specifically this means that *foo@.service* will never be shown in this list — unless instantiated, e.g. as *foo@bar.service*. Use **list-unit-files** (see below) for listing installed unit template files.

Produces output similar to

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
sys-module-fuse.device	loaded	active	plugged	/sys/module/fuse
-.mount	loaded	active	mounted	Root Mount
boot-efi.mount	loaded	active	mounted	/boot/efi
systemd-journald.service	loaded	active	running	Journal Service
systemd-logind.service	loaded	active	running	Login Service
user@1000.service	loaded	failed	failed	User Manager for UID 1000
...				
systemd-tmpfiles-clean.timer	loaded	active	waiting	Daily Cleanup of Temporary Directories

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of **SUB**.

SUB = The low-level unit activation state, values depend on unit type.

123 loaded units listed. Pass **--all** to see loaded but inactive units, too.

To show all installed unit files use '**systemctl list-unit-files**'.

The header and the last unit of a given type are underlined if the terminal supports that. A colored dot is shown next to services which were masked, not found, or otherwise failed.

The LOAD column shows the load state, one of **loaded**, **not-found**, **bad-setting**, **error**, **masked**. The ACTIVE columns shows the general unit state, one of **active**, **reloading**, **inactive**, **failed**, **activating**, **deactivating**. The SUB column shows the unit-type-specific detailed state of the unit, possible values vary by unit type. The list of possible LOAD, ACTIVE, and SUB states is not constant and new systemd releases may both add and remove values.

systemctl --state=help

command maybe be used to display the current set of possible values.

This is the default command.

list-sockets [PATTERN...]

List socket units currently in memory, ordered by listening address. If one or more *PATTERNs* are specified, only socket units matching one of them are shown. Produces output similar to

LISTEN	UNIT	ACTIVATES
/dev/initctl	systemd-initctl.socket	systemd-initctl.service
...		
[::]:22	sshd.socket	sshd.service
kobject-uevent 1	systemd-udevd-kernel.socket	systemd-udevd.service

5 sockets listed.

Note: because the addresses might contains spaces, this output is not suitable for programmatic consumption.

Also see **--show-types**, **--all**, and **--state=**.

list-timers [PATTERN...]

List timer units currently in memory, ordered by the time they elapse next. If one or more *PATTERNs* are specified, only units matching one of them are shown. Produces output similar to

NEXT	LEFT	LAST	PASSED	UNIT	ACTIVATES
n/a	n/a	Thu 2017-02-23 13:40:29 EST	3 days ago	ureadahead-stop.timer	ureadahead-stop.
Sun 2017-02-26 18:55:42 EST	1min 14s left	Thu 2017-02-23 13:54:44 EST	3 days ago	systemd-tmpfiles-clean.	
Sun 2017-02-26 20:37:16 EST	1h 42min left	Sun 2017-02-26 11:56:36 EST	6h ago	apt-daily.timer	ap
Sun 2017-02-26 20:57:49 EST	2h 3min left	Sun 2017-02-26 11:56:36 EST	6h ago	snapd.refresh.timer	sr

NEXT shows the next time the timer will run.

LEFT shows how long till the next time the timer runs.

LAST shows the last time the timer ran.

PASSED shows how long has passed since the timer last ran.

UNIT shows the name of the timer

ACTIVATES shows the name the service the timer activates when it runs.

Also see **--all** and **--state=**.

is-active PATTERN...

Check whether any of the specified units are active (i.e. running). Returns an exit code **0** if at least one is active, or non-zero otherwise. Unless **--quiet** is specified, this will also print the current unit state to standard output.

is-failed PATTERN...

Check whether any of the specified units are in a "failed" state. Returns an exit code **0** if at least one has failed, non-zero otherwise. Unless **--quiet** is specified, this will also print the current unit state to standard output.

status [PATTERN...|PID...]]

Show terse runtime status information about one or more units, followed by most recent log data from the journal. If no units are specified, show system status. If combined with **--all**, also show the status of all units (subject to limitations specified with **-t**). If a PID is passed, show information about the unit the process belongs to.

This function is intended to generate human-readable output. If you are looking for computer-parsable output, use **show** instead. By default, this function only shows 10 lines of output and ellipsizes lines to fit in the terminal window. This can be changed with **--lines** and **--full**, see above. In addition, **journalctl --unit=NAME** or **journalctl --user-unit=NAME** use a similar filter for messages and might be more convenient.

systemd implicitly loads units as necessary, so just running the **status** will attempt to load a file. The command is thus not useful for determining if something was already loaded or not. The units may possibly also be quickly unloaded after the operation is completed if there's no reason to keep it in memory thereafter.

Example 1. Example output from systemctl status

```
$ systemctl status bluetooth
bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2017-01-04 13:54:04 EST; 1 weeks 0 days ago
       Docs: man:bluetoothd(8)
    Main PID: 930 (bluetoothd)
      Status: "Running"
        Tasks: 1
     Memory: 648.0K
        CPU: 435ms
      CGroup: /system.slice/bluetooth.service
              930 /usr/lib/bluetooth/bluetoothd
```

```
Jan 12 10:46:45 example.com bluetoothd[8900]: Not enough free handles to register service
Jan 12 10:46:45 example.com bluetoothd[8900]: Current Time Service could not be registered
Jan 12 10:46:45 example.com bluetoothd[8900]: gatt-time-server: Input/output error (5)
```

The dot ("") uses color on supported terminals to summarize the unit state at a glance. Along with its color, its shape varies according to its state: "inactive" or "maintenance" is a white circle ("○"), "active" is a green dot ("."), "deactivating" is a white dot, "failed" or "error" is a red cross ("×"), and "reloading" is a green clockwise circle arrow ("").

The "Loaded:" line in the output will show "loaded" if the unit has been loaded into memory. Other possible values for "Loaded:" include: "error" if there was a problem loading it, "not-found" if no unit file was found for this unit, "bad-setting" if an essential unit file setting could not be parsed and "masked" if the unit file has been masked. Along with showing the path to the unit file, this line will also show the enablement state. Enabled commands start at boot. See the full table of possible enablement states — including the definition of "masked" — in the documentation for the **is-enabled** command.

The "Active:" line shows active state. The value is usually "active" or "inactive". Active could mean started, bound, plugged in, etc depending on the unit type. The unit could also be in process of changing states, reporting a state of "activating" or "deactivating". A special "failed" state is entered when the service failed in some way, such as a crash, exiting with an error code or timing out. If the failed state is entered the cause will be logged for later reference.

show [PATTERN...|JOB...]

Show properties of one or more units, jobs, or the manager itself. If no argument is specified,

properties of the manager will be shown. If a unit name is specified, properties of the unit are shown, and if a job ID is specified, properties of the job are shown. By default, empty properties are suppressed. Use **--all** to show those too. To select specific properties to show, use **--property=**. This command is intended to be used whenever computer-parsable output is required. Use **status** if you are looking for formatted human-readable output.

Many properties shown by **systemctl show** map directly to configuration settings of the system and service manager and its unit files. Note that the properties shown by the command are generally more low-level, normalized versions of the original configuration settings and expose runtime state in addition to configuration. For example, properties shown for service units include the service's current main process identifier as "MainPID" (which is runtime state), and time settings are always exposed as properties ending in the "...USec" suffix even if a matching configuration options end in "...Sec", because microseconds is the normalized time unit used internally by the system and service manager.

For details about many of these properties, see the documentation of the D-Bus interface backing these properties, see [org.freedesktop.systemd1\(5\)](#).

cat *PATTERN...*

Show backing files of one or more units. Prints the "fragment" and "drop-ins" (source files) of units. Each file is preceded by a comment which includes the file name. Note that this shows the contents of the backing files on disk, which may not match the system manager's understanding of these units if any unit files were updated on disk and the **daemon-reload** command wasn't issued since.

help *PATTERN...|PID...*

Show manual pages for one or more units, if available. If a PID is given, the manual pages for the unit the process belongs to are shown.

list-dependencies [*UNIT...*]

Shows units required and wanted by the specified units. This recursively lists units following the *Requires=*, *Requisite=*, *ConsistsOf=*, *Wants=*, *BindsTo=* dependencies. If no units are specified, *default.target* is implied.

By default, only target units are recursively expanded. When **--all** is passed, all other units are recursively expanded as well.

Options **--reverse**, **--after**, **--before** may be used to change what types of dependencies are shown.

Note that this command only lists units currently loaded into memory by the service manager. In particular, this command is not suitable to get a comprehensive list at all reverse dependencies on a specific unit, as it won't list the dependencies declared by units currently not loaded.

start *PATTERN...*

Start (activate) one or more units specified on the command line.

Note that unit glob patterns expand to names of units currently in memory. Units which are not active and are not in a failed state usually are not in memory, and will not be matched by any pattern. In addition, in case of instantiated units, systemd is often unaware of the instance name until the instance has been started. Therefore, using glob patterns with **start** has limited usefulness. Also, secondary alias names of units are not considered.

Option **--all** may be used to also operate on inactive units which are referenced by other loaded units. Note that this is not the same as operating on "all" possible units, because as the previous paragraph describes, such a list is ill-defined. Nevertheless, **systemctl start --all GLOB** may be useful if all the units that should match the pattern are pulled in by some target which is known to be loaded.

stop *PATTERN...*

Stop (deactivate) one or more units specified on the command line.

This command will fail if the unit does not exist or if stopping of the unit is prohibited (see `RefuseManualStop=` in [systemd.unit\(5\)](#)). It will *not* fail if any of the commands configured to stop the unit (`ExecStop=`, etc.) fail, because the manager will still forcibly terminate the unit.

reload PATTERN...

Asks all units listed on the command line to reload their configuration. Note that this will reload the service-specific configuration, not the unit configuration file of systemd. If you want systemd to reload the configuration file of a unit, use the **daemon-reload** command. In other words: for the example case of Apache, this will reload Apache's httpd.conf in the web server, not the apache.service systemd unit file.

This command should not be confused with the **daemon-reload** command.

restart PATTERN...

Stop and then start one or more units specified on the command line. If the units are not running yet, they will be started.

Note that restarting a unit with this command does not necessarily flush out all of the unit's resources before it is started again. For example, the per-service file descriptor storage facility (see `FileDescriptorStoreMax=` in [systemd.service\(5\)](#)) will remain intact as long as the unit has a job pending, and is only cleared when the unit is fully stopped and no jobs are pending anymore. If it is intended that the file descriptor store is flushed out, too, during a restart operation an explicit **systemctl stop** command followed by **systemctl start** should be issued.

try-restart PATTERN...

Stop and then start one or more units specified on the command line if the units are running. This does nothing if units are not running.

reload-or-restart PATTERN...

Reload one or more units if they support it. If not, stop and then start them instead. If the units are not running yet, they will be started.

try-reload-or-restart PATTERN...

Reload one or more units if they support it. If not, stop and then start them instead. This does nothing if the units are not running.

isolate UNIT

Start the unit specified on the command line and its dependencies and stop all others, unless they have `IgnoreOnIsolate=yes` (see [systemd.unit\(5\)](#)). If a unit name with no extension is given, an extension of ".target" will be assumed.

This command is dangerous, since it will immediately stop processes that are not enabled in the new target, possibly including the graphical environment or terminal you are currently using.

Note that this is allowed only on units where `AllowIsolate=` is enabled. See [systemd.unit\(5\)](#) for details.

kill PATTERN...

Send a signal to one or more processes of the unit. Use `--kill-who=` to select which process to kill. Use `--signal=` to select the signal to send.

clean PATTERN...

Remove the configuration, state, cache, logs or runtime data of the specified units. Use `--what=` to select which kind of resource to remove. For service units this may be used to remove the directories configured with `ConfigurationDirectory=`, `StateDirectory=`, `CacheDirectory=`, `LogsDirectory=` and `RuntimeDirectory=`, see [systemd.exec\(5\)](#) for details. For timer units this may be used to clear out the persistent timestamp data if `Persistent=` is used and `--what=state` is selected, see [systemd.timer\(5\)](#). This command only applies to units that use either of these settings. If `--what=` is not specified, both the cache and runtime data are removed (as these two types of data are generally redundant and

reproducible on the next invocation of the unit).

freeze PATTERN...

Freeze one or more units specified on the command line using cgroup freezer

Freezing the unit will cause all processes contained within the cgroup corresponding to the unit to be suspended. Being suspended means that unit's processes won't be scheduled to run on CPU until thawed. Note that this command is supported only on systems that use unified cgroup hierarchy. Unit is automatically thawed just before we execute a job against the unit, e.g. before the unit is stopped.

thaw PATTERN...

Thaw (unfreeze) one or more units specified on the command line.

This is the inverse operation to the **freeze** command and resumes the execution of processes in the unit's cgroup.

set–property UNIT PROPERTY=VALUE...

Set the specified unit properties at runtime where this is supported. This allows changing configuration parameter properties such as resource control settings at runtime. Not all properties may be changed at runtime, but many resource control settings (primarily those in **systemd.resource-control(5)**) may. The changes are applied immediately, and stored on disk for future boots, unless **--runtime** is passed, in which case the settings only apply until the next reboot. The syntax of the property assignment follows closely the syntax of assignments in unit files.

Example: **systemctl set–property foobar.service CPUWeight=200**

If the specified unit appears to be inactive, the changes will be only stored on disk as described previously hence they will be effective when the unit will be started.

Note that this command allows changing multiple properties at the same time, which is preferable over setting them individually.

Example: **systemctl set–property foobar.service CPUWeight=200 MemoryMax=2G IPAccounting=yes**

Like with unit file configuration settings, assigning an empty setting usually resets a property to its defaults.

Example: **systemctl set–property avahi–daemon.service IPAddressDeny=**

bind UNIT PATH [PATH]

Bind-mounts a file or directory from the host into the specified unit's mount namespace. The first path argument is the source file or directory on the host, the second path argument is the destination file or directory in the unit's mount namespace. When the latter is omitted, the destination path in the unit's mount namespace is the same as the source path on the host. When combined with the **--read-only** switch, a ready-only bind mount is created. When combined with the **--mkdir** switch, the destination path is first created before the mount is applied.

Note that this option is currently only supported for units that run within a mount namespace (e.g.: with **RootImage=**, **PrivateMounts=**, etc.). This command supports bind-mounting directories, regular files, device nodes, **AF_UNIX** socket nodes, as well as FIFOs. The bind mount is ephemeral, and it is undone as soon as the current unit process exists. Note that the namespace mentioned here, where the bind mount will be added to, is the one where the main service process runs. Other processes (those executed by **ExecReload=**, **ExecStartPre=**, etc.) run in distinct namespaces.

mount–image UNIT IMAGE [PATH [PARTITION_NAME:MOUNT_OPTIONS]]

Mounts an image from the host into the specified unit's mount namespace. The first path argument is

the source image on the host, the second path argument is the destination directory in the unit's mount namespace (i.e. inside **RootImage=/RootDirectory=**). The following argument, if any, is interpreted as a colon-separated tuple of partition name and comma-separated list of mount options for that partition. The format is the same as the service **MountImages=** setting. When combined with the **--read-only** switch, a ready-only mount is created. When combined with the **--mkdir** switch, the destination path is first created before the mount is applied.

Note that this option is currently only supported for units that run within a mount namespace (i.e. with **RootImage=**, **PrivateMounts=**, etc.). Note that the namespace mentioned here where the image mount will be added to, is the one where the main service process runs. Note that the namespace mentioned here, where the bind mount will be added to, is the one where the main service process runs. Other processes (those executed by **ExecReload=**, **ExecStartPre=**, etc.) run in distinct namespaces.

Example:

```
systemctl mount-image foo.service /tmp/img.raw /var/lib/image root:ro,nosuid
```

```
systemctl mount-image --mkdir bar.service /tmp/img.raw /var/lib/baz/img
```

service-log-level *SERVICE [LEVEL]*

If the *LEVEL* argument is not given, print the current log level as reported by service *SERVICE*.

If the optional argument *LEVEL* is provided, then change the current log level of the service to *LEVEL*. The log level should be a typical syslog log level, i.e. a value in the range 0...7 or one of the strings **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**; see **syslog(3)** for details.

The service must have the appropriate *BusName=destination* property and also implement the generic **org.freedesktop.LogControl1(5)** interface. (systemctl will use the generic D-Bus protocol to access the **org.freedesktop.LogControl1.LogLevel** interface for the D-Bus name *destination*.)

service-log-target *SERVICE [TARGET]*

If the *TARGET* argument is not given, print the current log target as reported by service *SERVICE*.

If the optional argument *TARGET* is provided, then change the current log target of the service to *TARGET*. The log target should be one of the strings **console** (for log output to the service's standard error stream), **kmsg** (for log output to the kernel log buffer), **journal** (for log output to **systemd-journald.service(8)** using the native journal protocol), **syslog** (for log output to the classic syslog socket /dev/log), **null** (for no log output whatsoever) or **auto** (for an automatically determined choice, typically equivalent to **console** if the service is invoked interactively, and **journal** or **syslog** otherwise).

For most services, only a small subset of log targets make sense. In particular, most "normal" services should only implement **console**, **journal**, and **null**. Anything else is only appropriate for low-level services that are active in very early boot before proper logging is established.

The service must have the appropriate *BusName=destination* property and also implement the generic **org.freedesktop.LogControl1(5)** interface. (systemctl will use the generic D-Bus protocol to access the **org.freedesktop.LogControl1.LogLevel** interface for the D-Bus name *destination*.)

reset-failed [*PATTERN...*]

Reset the "failed" state of the specified units, or if no unit name is passed, reset the state of all units. When a unit fails in some way (i.e. process exiting with non-zero error code, terminating abnormally or timing out), it will automatically enter the "failed" state and its exit code and status is recorded for

introspection by the administrator until the service is stopped/re-started or reset with this command.

In addition to resetting the "failed" state of a unit it also resets various other per-unit properties: the start rate limit counter of all unit types is reset to zero, as is the restart counter of service units. Thus, if a unit's start limit (as configured with *StartLimitIntervalSec=/StartLimitBurst=*) is hit and the unit refuses to be started again, use this command to make it startable again.

Unit File Commands

list-unit-files [*PATTERN...*]

List unit files installed on the system, in combination with their enablement state (as reported by **is-enabled**). If one or more *PATTERNs* are specified, only unit files whose name matches one of them are shown (patterns matching unit file system paths are not supported).

Unlike **list-units** this command will list template units in addition to explicitly instantiated units.

enable *UNIT...*, **enable** *PATH...*

Enable one or more units or unit instances. This will create a set of symlinks, as encoded in the [Install] sections of the indicated unit files. After the symlinks have been created, the system manager configuration is reloaded (in a way equivalent to **daemon-reload**), in order to ensure the changes are taken into account immediately. Note that this does *not* have the effect of also starting any of the units being enabled. If this is desired, combine this command with the **--now** switch, or invoke **start** with appropriate arguments later. Note that in case of unit instance enablement (i.e. enablement of units of the form *foo@bar.service*), symlinks named the same as instances are created in the unit configuration directory, however they point to the single template unit file they are instantiated from.

This command expects either valid unit names (in which case various unit file directories are automatically searched for unit files with appropriate names), or absolute paths to unit files (in which case these files are read directly). If a specified unit file is located outside of the usual unit file directories, an additional symlink is created, linking it into the unit configuration path, thus ensuring it is found when requested by commands such as **start**. The file system where the linked unit files are located must be accessible when systemd is started (e.g. anything underneath /home/ or /var/ is not allowed, unless those directories are located on the root file system).

This command will print the file system operations executed. This output may be suppressed by passing **--quiet**.

Note that this operation creates only the symlinks suggested in the [Install] section of the unit files. While this command is the recommended way to manipulate the unit configuration directory, the administrator is free to make additional changes manually by placing or removing symlinks below this directory. This is particularly useful to create configurations that deviate from the suggested default installation. In this case, the administrator must make sure to invoke **daemon-reload** manually as necessary, in order to ensure the changes are taken into account.

Enabling units should not be confused with starting (activating) units, as done by the **start** command. Enabling and starting units is orthogonal: units may be enabled without being started and started without being enabled. Enabling simply hooks the unit into various suggested places (for example, so that the unit is automatically started on boot or when a particular kind of hardware is plugged in). Starting actually spawns the daemon process (in case of service units), or binds the socket (in case of socket units), and so on.

Depending on whether **--system**, **--user**, **--runtime**, or **--global** is specified, this enables the unit for the system, for the calling user only, for only this boot of the system, or for all future logins of all users. Note that in the last case, no systemd daemon configuration is reloaded.

Using **enable** on masked units is not supported and results in an error.

disable *UNIT...*

Disables one or more units. This removes all symlinks to the unit files backing the specified units from the unit configuration directory, and hence undoes any changes made by **enable** or **link**. Note that this removes *all* symlinks to matching unit files, including manually created symlinks, and not just those actually created by **enable** or **link**. Note that while **disable** undoes the effect of **enable**, the two commands are otherwise not symmetric, as **disable** may remove more symlinks than a prior **enable** invocation of the same unit created.

This command expects valid unit names only, it does not accept paths to unit files.

In addition to the units specified as arguments, all units are disabled that are listed in the *Also=* setting contained in the [Install] section of any of the unit files being operated on.

This command implicitly reloads the system manager configuration after completing the operation. Note that this command does not implicitly stop the units that are being disabled. If this is desired, either combine this command with the **--now** switch, or invoke the **stop** command with appropriate arguments later.

This command will print information about the file system operations (symlink removals) executed. This output may be suppressed by passing **--quiet**.

This command honors **--system**, **--user**, **--runtime** and **--global** in a similar way as **enable**.

reenable *UNIT...*

Reenable one or more units, as specified on the command line. This is a combination of **disable** and **enable** and is useful to reset the symlinks a unit file is enabled with to the defaults configured in its [Install] section. This command expects a unit name only, it does not accept paths to unit files.

preset *UNIT...*

Reset the enable/disable status one or more unit files, as specified on the command line, to the defaults configured in the preset policy files. This has the same effect as **disable** or **enable**, depending how the unit is listed in the preset files.

Use **--preset-mode=** to control whether units shall be enabled and disabled, or only enabled, or only disabled.

If the unit carries no install information, it will be silently ignored by this command. *UNIT* must be the real unit name, any alias names are ignored silently.

For more information on the preset policy format, see **systemd.preset(5)**.

preset-all

Resets all installed unit files to the defaults configured in the preset policy file (see above).

Use **--preset-mode=** to control whether units shall be enabled and disabled, or only enabled, or only disabled.

is-enabled *UNIT...*

Checks whether any of the specified unit files are enabled (as with **enable**). Returns an exit code of 0 if at least one is enabled, non-zero otherwise. Prints the current enable status (see table). To suppress this output, use **--quiet**. To show installation targets, use **--full**.

Table 1. is-enabled output

mask *UNIT...*

Mask one or more units, as specified on the command line. This will link these unit files to /dev/null, making it impossible to start them. This is a stronger version of **disable**, since it prohibits all kinds of activation of the unit, including enablement and manual activation. Use this option with care. This honors the **--runtime** option to only mask temporarily until the next reboot of the system. The **--now** option may be used to ensure that the units are also stopped. This command expects valid unit names only, it does not accept unit file paths.

unmask *UNIT...*

Unmask one or more unit files, as specified on the command line. This will undo the effect of **mask**. This command expects valid unit names only, it does not accept unit file paths.

link *PATH...*

Link a unit file that is not in the unit file search paths into the unit file search path. This command expects an absolute path to a unit file. The effect of this may be undone with **disable**. The effect of this command is that a unit file is made available for commands such as **start**, even though it is not installed directly in the unit search path. The file system where the linked unit files are located must be accessible when systemd is started (e.g. anything underneath /home/ or /var/ is not allowed, unless those directories are located on the root file system).

revert *UNIT...*

Revert one or more unit files to their vendor versions. This command removes drop-in configuration files that modify the specified units, as well as any user-configured unit file that overrides a matching vendor supplied unit file. Specifically, for a unit "foo.service" the matching directories "foo.service.d/" with all their contained files are removed, both below the persistent and runtime configuration directories (i.e. below /etc/systemd/system and /run/systemd/system); if the unit file has a vendor-supplied version (i.e. a unit file located below /usr/) any matching persistent or runtime unit file that overrides it is removed, too. Note that if a unit file has no vendor-supplied version (i.e. is only defined below /etc/systemd/system or /run/systemd/system, but not in a unit file stored below /usr/), then it is not removed. Also, if a unit is masked, it is unmasked.

Effectively, this command may be used to undo all changes made with **systemctl edit**, **systemctl set-property** and **systemctl mask** and puts the original unit file with its settings back in effect.

add-wants *TARGET UNIT...*, **add-requires** *TARGET UNIT...*

Adds "Wants=" or "Requires=" dependencies, respectively, to the specified *TARGET* for one or more units.

This command honors **--system**, **--user**, **--runtime** and **--global** in a way similar to **enable**.

edit *UNIT...*

Edit a drop-in snippet or a whole replacement file if **--full** is specified, to extend or override the specified unit.

Depending on whether **--system** (the default), **--user**, or **--global** is specified, this command creates a drop-in file for each unit either for the system, for the calling user, or for all future logins of all users. Then, the editor (see the "Environment" section below) is invoked on temporary files which will be written to the real location if the editor exits successfully.

If **--full** is specified, this will copy the original units instead of creating drop-in files.

If **--force** is specified and any units do not already exist, new unit files will be opened for editing.

If **--runtime** is specified, the changes will be made temporarily in /run/ and they will be lost on the next reboot.

If the temporary file is empty upon exit, the modification of the related unit is canceled.

After the units have been edited, systemd configuration is reloaded (in a way that is equivalent to **daemon-reload**).

Note that this command cannot be used to remotely edit units and that you cannot temporarily edit units which are in /etc/, since they take precedence over /run/.

get-default

Return the default target to boot into. This returns the target unit name default.target is aliased (symlinked) to.

set-default TARGET

Set the default target to boot into. This sets (symlinks) the default.target alias to the given target unit.

Machine Commands

list-machines [PATTERN...]

List the host and all running local containers with their state. If one or more *PATTERNs* are specified, only containers matching one of them are shown.

Job Commands

list-jobs [PATTERN...]

List jobs that are in progress. If one or more *PATTERNs* are specified, only jobs for units matching one of them are shown.

When combined with **--after** or **--before** the list is augmented with information on which other job each job is waiting for, and which other jobs are waiting for it, see above.

cancel JOB...

Cancel one or more jobs specified on the command line by their numeric job IDs. If no job ID is specified, cancel all pending jobs.

Environment Commands

systemd supports an environment block that is passed to processes the manager spawns. The names of the variables can contain ASCII letters, digits, and the underscore character. Variable names cannot be empty or start with a digit. In variable values, most characters are allowed, but the whole sequence must be valid UTF-8. (Note that control characters like newline (**NL**), tab (**TAB**), or the escape character (**ESC**), are valid ASCII and thus valid UTF-8). The total length of the environment block is limited to **_SC_ARG_MAX** value defined by **sysconf(3)**.

show-environment

Dump the systemd manager environment block. This is the environment block that is passed to all processes the manager spawns. The environment block will be dumped in straight-forward form suitable for sourcing into most shells. If no special characters or whitespace is present in the variable values, no escaping is performed, and the assignments have the form "VARIABLE=value". If whitespace or characters which have special meaning to the shell are present, dollar-single-quote escaping is used, and assignments have the form "VARIABLE=\$'value'". This syntax is known to be supported by **bash(1)**, **zsh(1)**, **ksh(1)**, and **busybox(1)**'s **ash(1)**, but not **dash(1)** or **fish(1)**.

set-environment VARIABLE=VALUE...

Set one or more systemd manager environment variables, as specified on the command line. This command will fail if variable names and values do not conform to the rules listed above.

unset-environment VARIABLE...

Unset one or more systemd manager environment variables. If only a variable name is specified, it will be removed regardless of its value. If a variable and a value are specified, the variable is only removed if it has the specified value.

import-environment VARIABLE...

Import all, one or more environment variables set on the client into the systemd manager environment block. If a list of environment variable names is passed, client-side values are then imported into the manager's environment block. If any names are not valid environment variable names or have invalid

values according to the rules described above, an error is raised. If no arguments are passed, the entire environment block inherited by the **systemctl** process is imported. In this mode, any inherited invalid environment variables are quietly ignored.

Importing of the full inherited environment block (calling this command without any arguments) is deprecated. A shell will set dozens of variables which only make sense locally and are only meant for processes which are descendants of the shell. Such variables in the global environment block are confusing to other processes.

Manager State Commands

daemon-reload

Reload the systemd manager configuration. This will rerun all generators (see **systemd.generator(7)**), reload all unit files, and recreate the entire dependency tree. While the daemon is being reloaded, all sockets systemd listens on behalf of user configuration will stay accessible.

This command should not be confused with the **reload** command.

daemon-reeexec

Reexecute the systemd manager. This will serialize the manager state, reexecute the process and deserialize the state again. This command is of little use except for debugging and package upgrades. Sometimes, it might be helpful as a heavy-weight **daemon-reload**. While the daemon is being reexecuted, all sockets systemd listening on behalf of user configuration will stay accessible.

log-level [LEVEL]

If no argument is given, print the current log level of the manager. If an optional argument *LEVEL* is provided, then the command changes the current log level of the manager to *LEVEL* (accepts the same values as **--log-level=** described in **systemd(1)**).

log-target [TARGET]

If no argument is given, print the current log target of the manager. If an optional argument *TARGET* is provided, then the command changes the current log target of the manager to *TARGET* (accepts the same values as **--log-target=**, described in **systemd(1)**).

service-watchdogs [yes|no]

If no argument is given, print the current state of service runtime watchdogs of the manager. If an optional boolean argument is provided, then globally enables or disables the service runtime watchdogs (**WatchdogSec=**) and emergency actions (e.g. **OnFailure=** or **StartLimitAction=**); see **systemd.service(5)**. The hardware watchdog is not affected by this setting.

System Commands

is-system-running

Checks whether the system is operational. This returns success (exit code 0) when the system is fully up and running, specifically not in startup, shutdown or maintenance mode, and with no failed services. Failure is returned otherwise (exit code non-zero). In addition, the current state is printed in a short string to standard output, see the table below. Use **--quiet** to suppress this output.

Use **--wait** to wait until the boot process is completed before printing the current state and returning the appropriate error status. If **--wait** is in use, states *initializing* or *starting* will not be reported, instead the command will block until a later state (such as *running* or *degraded*) is reached.

Table 2. is-system-running output

Name	Description	Exit Code
<i>initializing</i>	Early bootup, before basic.target is reached or the <i>maintenance</i> state entered.	> 0
<i>starting</i>	Late bootup, before the job queue becomes idle for the first time, or one of the rescue targets are reached.	> 0
<i>running</i>	The system is fully operational.	0
<i>degraded</i>	The system is operational but one or more units failed.	> 0
<i>maintenance</i>	The rescue or emergency target is active.	> 0
<i>stopping</i>	The manager is shutting down.	> 0
<i>offline</i>	The manager is not running. Specifically, this is the operational state if an incompatible program is running as system manager (PID 1).	> 0
<i>unknown</i>	The operational state could not be determined, due to lack of resources or another error cause.	> 0

default

Enter default mode. This is equivalent to **systemctl isolate default.target**. This operation is blocking by default, use **--no-block** to request asynchronous behavior.

rescue

Enter rescue mode. This is equivalent to **systemctl isolate rescue.target**. This operation is blocking by default, use **--no-block** to request asynchronous behavior.

emergency

Enter emergency mode. This is equivalent to **systemctl isolate emergency.target**. This operation is blocking by default, use **--no-block** to request asynchronous behavior.

halt

Shut down and halt the system. This is mostly equivalent to **systemctl start halt.target --job-mode=replace-irreversibly --no-block**, but also prints a wall message to all users. This command is asynchronous; it will return after the halt operation is enqueued, without waiting for it to complete. Note that this operation will simply halt the OS kernel after shutting down, leaving the hardware powered on. Use **systemctl poweroff** for powering off the system (see below).

If combined with **--force**, shutdown of all running services is skipped, however all processes are killed and all file systems are unmounted or mounted read-only, immediately followed by the system halt. If **--force** is specified twice, the operation is immediately executed without terminating any processes or unmounting any file systems. This may result in data loss. Note that when **--force** is specified twice the halt operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

poweroff

Shut down and power-off the system. This is mostly equivalent to **systemctl start poweroff.target**

--job-mode=replace-irreversibly --no-block, but also prints a wall message to all users. This command is asynchronous; it will return after the power-off operation is enqueued, without waiting for it to complete.

If combined with **--force**, shutdown of all running services is skipped, however all processes are killed and all file systems are unmounted or mounted read-only, immediately followed by the powering off. If **--force** is specified twice, the operation is immediately executed without terminating any processes or unmounting any file systems. This may result in data loss. Note that when **--force** is specified twice the power-off operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

reboot

Shut down and reboot the system. This is mostly equivalent to **systemctl start reboot.target** **--job-mode=replace-irreversibly --no-block**, but also prints a wall message to all users. This command is asynchronous; it will return after the reboot operation is enqueued, without waiting for it to complete.

If combined with **--force**, shutdown of all running services is skipped, however all processes are killed and all file systems are unmounted or mounted read-only, immediately followed by the reboot. If **--force** is specified twice, the operation is immediately executed without terminating any processes or unmounting any file systems. This may result in data loss. Note that when **--force** is specified twice the reboot operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

If the switch **--reboot-argument=** is given, it will be passed as the optional argument to the **reboot(2)** system call.

kexec

Shut down and reboot the system via **kexec**. This is equivalent to **systemctl start kexec.target** **--job-mode=replace-irreversibly --no-block**. This command is asynchronous; it will return after the reboot operation is enqueued, without waiting for it to complete.

If combined with **--force**, shutdown of all running services is skipped, however all processes are killed and all file systems are unmounted or mounted read-only, immediately followed by the reboot.

exit [*EXIT_CODE*]

Ask the service manager to quit. This is only supported for user service managers (i.e. in conjunction with the **--user** option) or in containers and is equivalent to **poweroff** otherwise. This command is asynchronous; it will return after the exit operation is enqueued, without waiting for it to complete.

The service manager will exit with the specified exit code, if *EXIT_CODE* is passed.

switch-root *ROOT* [*INIT*]

Switches to a different root directory and executes a new system manager process below it. This is intended for usage in initial RAM disks ("initrd"), and will transition from the initrd's system manager process (a.k.a. "init" process) to the main system manager process which is loaded from the actual host volume. This call takes two arguments: the directory that is to become the new root directory, and the path to the new system manager binary below it to execute as PID 1. If the latter is omitted or the empty string, a systemd binary will automatically be searched for and used as init. If the system manager path is omitted, equal to the empty string or identical to the path to the systemd binary, the state of the initrd's system manager process is passed to the main system manager, which allows later introspection of the state of the services involved in the initrd boot phase.

suspend

Suspend the system. This will trigger activation of the special target unit **suspend.target**. This command is asynchronous, and will return after the suspend operation is successfully enqueued. It will not wait for the suspend/resume cycle to complete.

hibernate

Hibernate the system. This will trigger activation of the special target unit hibernate.target. This command is asynchronous, and will return after the hibernation operation is successfully enqueued. It will not wait for the hibernate/thaw cycle to complete.

hybrid-sleep

Hibernate and suspend the system. This will trigger activation of the special target unit hybrid-sleep.target. This command is asynchronous, and will return after the hybrid sleep operation is successfully enqueued. It will not wait for the sleep/wake-up cycle to complete.

suspend-then-hibernate

Suspend the system and hibernate it after the delay specified in systemd-sleep.conf. This will trigger activation of the special target unit suspend-then-hibernate.target. This command is asynchronous, and will return after the hybrid sleep operation is successfully enqueued. It will not wait for the sleep/wake-up or hibernate/thaw cycle to complete.

Parameter Syntax

Unit commands listed above take either a single unit name (designated as *UNIT*), or multiple unit specifications (designated as *PATTERN...*). In the first case, the unit name with or without a suffix must be given. If the suffix is not specified (unit name is "abbreviated"), systemctl will append a suitable suffix, ".service" by default, and a type-specific suffix in case of commands which operate only on specific unit types. For example,

```
# systemctl start sshd
```

and

```
# systemctl start sshd.service
```

are equivalent, as are

```
# systemctl isolate default
```

and

```
# systemctl isolate default.target
```

Note that (absolute) paths to device nodes are automatically converted to device unit names, and other (absolute) paths to mount unit names.

```
# systemctl status /dev/sda
# systemctl status /home
```

are equivalent to:

```
# systemctl status dev-sda.device
# systemctl status home.mount
```

In the second case, shell-style globs will be matched against the primary names of all units currently in memory; literal unit names, with or without a suffix, will be treated as in the first case. This means that literal unit names always refer to exactly one unit, but globs may match zero units and this is not considered an error.

Glob patterns use **fnmatch(3)**, so normal shell-style globbing rules are used, and "*", "?", "[]" may be used. See **glob(7)** for more details. The patterns are matched against the primary names of units currently in memory, and patterns which do not match anything are silently skipped. For example:

```
# systemctl stop sshd@*.service
```

will stop all sshd@.service instances. Note that alias names of units, and units that aren't in memory are not considered for glob expansion.

For unit file commands, the specified *UNIT* should be the name of the unit file (possibly abbreviated, see above), or the absolute path to the unit file:

```
# systemctl enable foo.service
```

or

```
# systemctl link /path/to/foo.service
```

OPTIONS

The following options are understood:

-t, --type=

The argument should be a comma-separated list of unit types such as **service** and **socket**.

If one of the arguments is a unit type, when listing units, limit display to certain unit types. Otherwise, units of all types will be shown.

As a special case, if one of the arguments is **help**, a list of allowed values will be printed and the program will exit.

--state=

The argument should be a comma-separated list of unit LOAD, SUB, or ACTIVE states. When listing units, show only those in the specified states. Use **--state=failed** to show only failed units.

As a special case, if one of the arguments is **help**, a list of allowed values will be printed and the program will exit.

-p, --property=

When showing unit/job/manager properties with the **show** command, limit display to properties specified in the argument. The argument should be a comma-separated list of property names, such as "MainPID". Unless specified, all known properties are shown. If specified more than once, all properties with the specified names are shown. Shell completion is implemented for property names.

For the manager itself, **systemctl show** will show all available properties, most of which are derived or closely match the options described in **systemd-system.conf(5)**.

Properties for units vary by unit type, so showing any unit (even a non-existent one) is a way to list properties pertaining to this type. Similarly, showing any job will list properties pertaining to all jobs. Properties for units are documented in **systemd.unit(5)**, and the pages for individual unit types **systemd.service(5)**, **systemd.socket(5)**, etc.

-P

Equivalent to **--value --property=**, i.e. shows the value of the property without the property name or "**=**". Note that using **-P** once will also affect all properties listed with **-p/--property=**.

-a, --all

When listing units with **list-units**, also show inactive units and units which are following other units. When showing unit/job/manager properties, show all properties regardless whether they are set or not.

To list all units installed in the file system, use the **list-unit-files** command instead.

When listing units with **list-dependencies**, recursively show dependencies of all dependent units (by

default only dependencies of target units are shown).

When used with **status**, show journal messages in full, even if they include unprintable characters or are very long. By default, fields with unprintable characters are abbreviated as "blob data". (Note that the pager may escape unprintable characters again.)

-r, --recursive

When listing units, also show units of local containers. Units of local containers will be prefixed with the container name, separated by a single colon character ("!").

--reverse

Show reverse dependencies between units with **list-dependencies**, i.e. follow dependencies of type *WantedBy=*, *RequiredBy=*, *PartOf=*, *BoundBy=*, instead of *Wants=* and similar.

--after

With **list-dependencies**, show the units that are ordered before the specified unit. In other words, recursively list units following the *After=* dependency.

Note that any *After=* dependency is automatically mirrored to create a *Before=* dependency. Temporal dependencies may be specified explicitly, but are also created implicitly for units which are *WantedBy=* targets (see **systemd.target(5)**), and as a result of other directives (for example *RequiresMountsFor=*). Both explicitly and implicitly introduced dependencies are shown with **list-dependencies**.

When passed to the **list-jobs** command, for each printed job show which other jobs are waiting for it. May be combined with **--before** to show both the jobs waiting for each job as well as all jobs each job is waiting for.

--before

With **list-dependencies**, show the units that are ordered after the specified unit. In other words, recursively list units following the *Before=* dependency.

When passed to the **list-jobs** command, for each printed job show which other jobs it is waiting for. May be combined with **--after** to show both the jobs waiting for each job as well as all jobs each job is waiting for.

--with-dependencies

When used with **status**, **cat**, **list-units**, and **list-unit-files**, those commands print all specified units and the dependencies of those units.

Options **--reverse**, **--after**, **--before** may be used to change what types of dependencies are shown.

-l, --full

Do not ellipsize unit names, process tree entries, journal output, or truncate unit descriptions in the output of **status**, **list-units**, **list-jobs**, and **list-timers**.

Also, show installation targets in the output of **is-enabled**.

--value

When printing properties with **show**, only print the value, and skip the property name and "*=*". Also see option **-P** above.

--show-types

When showing sockets, show the type of the socket.

--job-mode

When queuing a new job, this option controls how to deal with already queued jobs. It takes one of "fail", "replace", "replace-irreversibly", "isolate", "ignore-dependencies", "ignore-requirements", "flush", or "triggering". Defaults to "replace", except when the **isolate** command is used which implies the "isolate" job mode.

If "fail" is specified and a requested operation conflicts with a pending job (more specifically: causes an already pending start job to be reversed into a stop job or vice versa), cause the operation to fail.

If "replace" (the default) is specified, any conflicting pending job will be replaced, as necessary.

If "replace-irreversibly" is specified, operate like "replace", but also mark the new jobs as irreversible. This prevents future conflicting transactions from replacing these jobs (or even being enqueued while the irreversible jobs are still pending). Irreversible jobs can still be cancelled using the **cancel** command. This job mode should be used on any transaction which pulls in shutdown.target.

"isolate" is only valid for start operations and causes all other units to be stopped when the specified unit is started. This mode is always used when the **isolate** command is used.

"flush" will cause all queued jobs to be canceled when the new job is enqueued.

If "ignore-dependencies" is specified, then all unit dependencies are ignored for this new job and the operation is executed immediately. If passed, no required units of the unit passed will be pulled in, and no ordering dependencies will be honored. This is mostly a debugging and rescue tool for the administrator and should not be used by applications.

"ignore-requirements" is similar to "ignore-dependencies", but only causes the requirement dependencies to be ignored, the ordering dependencies will still be honored.

"triggering" may only be used with **systemctl stop**. In this mode, the specified unit and any active units that trigger it are stopped. See the discussion of *Triggers=* in **systemd.unit(5)** for more information about triggering units.

-T, --show-transaction

When enqueueing a unit job (for example as effect of a **systemctl start** invocation or similar), show brief information about all jobs enqueued, covering both the requested job and any added because of unit dependencies. Note that the output will only include jobs immediately part of the transaction requested. It is possible that service start-up program code run as effect of the enqueued jobs might request further jobs to be pulled in. This means that completion of the listed jobs might ultimately entail more jobs than the listed ones.

--fail

Shorthand for **--job-mode=fail**.

When used with the **kill** command, if no units were killed, the operation results in an error.

--check-inhibitors=

When system shutdown or sleep state is request, this option controls how to deal with inhibitor locks. It takes one of "auto", "yes" or "no". Defaults to "auto", which will behave like "yes" for interactive invocations (i.e. from a TTY) and "no" for non-interactive invocations. "yes" will let the request respect inhibitor locks. "no" will let the request ignore inhibitor locks.

Applications can establish inhibitor locks to avoid that certain important operations (such as CD burning or suchlike) are interrupted by system shutdown or a sleep state. Any user may take these locks and privileged users may override these locks. If any locks are taken, shutdown and sleep state requests will normally fail (unless privileged) and a list of active locks is printed. However, if "no" is specified or "auto" is specified on a non-interactive requests, the established locks are ignored and not shown, and the operation attempted anyway, possibly requiring additional privileges. May be overridden by **--force**.

-i

Shortcut for **--check-inhibitors=no**.

--dry-run

Just print what would be done. Currently supported by verbs **halt**, **poweroff**, **reboot**, **kexec**, **suspend**, **hibernate**, **hybrid-sleep**, **suspend-then-hibernate**, **default**, **rescue**, **emergency**, and **exit**.

--q, --quiet

Suppress printing of the results of various commands and also the hints about truncated log lines. This does not suppress output of commands for which the printed output is the only result (like **show**). Errors are always printed.

--no-block

Do not synchronously wait for the requested operation to finish. If this is not specified, the job will be verified, enqueued and **systemctl** will wait until the unit's start-up is completed. By passing this argument, it is only verified and enqueued. This option may not be combined with **--wait**.

--wait

Synchronously wait for started units to terminate again. This option may not be combined with **--no-block**. Note that this will wait forever if any given unit never terminates (by itself or by getting stopped explicitly); particularly services which use "RemainAfterExit=yes".

When used with **is-system-running**, wait until the boot process is completed before returning.

--user

Talk to the service manager of the calling user, rather than the service manager of the system.

--system

Talk to the service manager of the system. This is the implied default.

--failed

List units in failed state. This is equivalent to **--state=failed**.

--no-wall

Do not send wall message before halt, power-off and reboot.

--global

When used with **enable** and **disable**, operate on the global user configuration directory, thus enabling or disabling a unit file globally for all future logins of all users.

--no-reload

When used with **enable** and **disable**, do not implicitly reload daemon configuration after executing the changes.

--no-ask-password

When used with **start** and related commands, disables asking for passwords. Background services may require input of a password or passphrase string, for example to unlock system hard disks or cryptographic certificates. Unless this option is specified and the command is invoked from a terminal, **systemctl** will query the user on the terminal for the necessary secrets. Use this option to switch this behavior off. In this case, the password must be supplied by some other means (for example graphical password agents) or the service might fail. This also disables querying the user for authentication for privileged operations.

--kill-who=

When used with **kill**, choose which processes to send a signal to. Must be one of **main**, **control** or **all** to select whether to kill only the main process, the control process or all processes of the unit. The main process of the unit is the one that defines the life-time of it. A control process of a unit is one that is invoked by the manager to induce state changes of it. For example, all processes started due to the *ExecStartPre=*, *ExecStop=* or *ExecReload=* settings of service units are control processes. Note that there is only one control process per unit at a time, as only one state change is executed at a time. For services of type *Type=forking*, the initial process started by the manager for *ExecStart=* is a control process, while the process ultimately forked off by that one is then considered the main process of the unit (if it can be determined). This is different for service units of other types, where the process forked off by the manager for *ExecStart=* is always the main process itself. A service unit consists of zero or one main process, zero or one control process plus any number of additional

processes. Not all unit types manage processes of these types however. For example, for mount units, control processes are defined (which are the invocations of `/bin/mount` and `/bin/umount`), but no main process is defined. If omitted, defaults to **all**.

-s, --signal=

When used with **kill**, choose which signal to send to selected processes. Must be one of the well-known signal specifiers such as **SIGTERM**, **SIGINT** or **SIGSTOP**. If omitted, defaults to **SIGTERM**.

The special value "help" will list the known values and the program will exit immediately, and the special value "list" will list known values along with the numerical signal numbers and the program will exit immediately.

--what=

Select what type of per-unit resources to remove when the **clean** command is invoked, see below. Takes one of **configuration**, **state**, **cache**, **logs**, **runtime** to select the type of resource. This option may be specified more than once, in which case all specified resource types are removed. Also accepts the special value **all** as a shortcut for specifying all five resource types. If this option is not specified defaults to the combination of **cache** and **runtime**, i.e. the two kinds of resources that are generally considered to be redundant and can be reconstructed on next invocation.

-f, --force

When used with **enable**, overwrite any existing conflicting symlinks.

When used with **edit**, create all of the specified units which do not already exist.

When used with **halt**, **poweroff**, **reboot** or **kexec**, execute the selected operation without shutting down all units. However, all processes will be killed forcibly and all file systems are unmounted or remounted read-only. This is hence a drastic but relatively safe option to request an immediate reboot. If **--force** is specified twice for these operations (with the exception of **kexec**), they will be executed immediately, without terminating any processes or unmounting any file systems. Warning: specifying **--force** twice with any of these operations might result in data loss. Note that when **--force** is specified twice the selected operation is executed by **systemctl** itself, and the system manager is not contacted. This means the command should succeed even when the system manager has crashed.

--message=

When used with **halt**, **poweroff** or **reboot**, set a short message explaining the reason for the operation. The message will be logged together with the default shutdown message.

--now

When used with **enable**, the units will also be started. When used with **disable** or **mask**, the units will also be stopped. The start or stop operation is only carried out when the respective enable or disable operation has been successful.

--root=

When used with **enable/disable/is-enabled** (and related commands), use the specified root path when looking for unit files. If this option is present, **systemctl** will operate on the file system directly, instead of communicating with the **systemd** daemon to carry out changes.

--runtime

When used with **enable**, **disable**, **edit**, (and related commands), make changes only temporarily, so that they are lost on the next reboot. This will have the effect that changes are not made in subdirectories of `/etc/` but in `/run/`, with identical immediate effects, however, since the latter is lost on reboot, the changes are lost too.

Similarly, when used with **set-property**, make changes only temporarily, so that they are lost on the next reboot.

--preset-mode=

Takes one of "full" (the default), "enable-only", "disable-only". When used with the **preset** or **preset-all** commands, controls whether units shall be disabled and enabled according to the preset rules, or only enabled, or only disabled.

-n, --lines=

When used with **status**, controls the number of journal lines to show, counting from the most recent ones. Takes a positive integer argument, or 0 to disable journal output. Defaults to 10.

-o, --output=

When used with **status**, controls the formatting of the journal entries that are shown. For the available choices, see **journalctl(1)**. Defaults to "short".

--firmware-setup

When used with the **reboot** command, indicate to the system's firmware to reboot into the firmware setup interface. Note that this functionality is not available on all systems.

--boot-loader-menu=

When used with the **reboot** command, indicate to the system's boot loader to show the boot loader menu on the following boot. Takes a time value as parameter — indicating the menu timeout. Pass zero in order to disable the menu timeout. Note that not all boot loaders support this functionality.

--boot-loader-entry=

When used with the **reboot** command, indicate to the system's boot loader to boot into a specific boot loader entry on the following boot. Takes a boot loader entry identifier as argument, or "help" in order to list available entries. Note that not all boot loaders support this functionality.

--reboot-argument=

This switch is used with **reboot**. The value is architecture and firmware specific. As an example, "recovery" might be used to trigger system recovery, and "fota" might be used to trigger a "firmware over the air" update.

--plain

When used with **list-dependencies**, **list-units** or **list-machines**, the output is printed as a list instead of a tree, and the bullet circles are omitted.

--timestamp=

Change the format of printed timestamps. The following values may be used:

pretty (this is the default)

"Day YYYY-MM-DD HH:MM:SS TZ"

us, us

"Day YYYY-MM-DD HH:MM:SS.UUUUUU TZ"

utc

"Day YYYY-MM-DD HH:MM:SS UTC"

us+utc, us+utc

"Day YYYY-MM-DD HH:MM:SS.UUUUUU UTC"

--mkdir

When used with **bind**, creates the destination file or directory before applying the bind mount. Note that even though the name of this option suggests that it is suitable only for directories, this option also creates the destination file node to mount over if the object to mount is not a directory, but a regular file, device node, socket or FIFO.

--marked

Only allowed with **reload-or-restart**. Enqueues restart jobs for all units that have the "needs-restart" mark, and reload jobs for units that have the "needs-reload" mark. When a unit marked for reload does not support reload, restart will be queued. Those properties can be set using **set-property Marks**.

Unless **--no-block** is used, **systemctl** will wait for the queued jobs to finish.

--read-only

When used with **bind**, creates a read-only bind mount.

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":"; and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

--no-pager

Do not pipe output into a pager.

--legend=BOOL

Enable or disable printing of the legend, i.e. column headers and the footer with hints. The legend is printed by default, unless disabled with **--quiet** or similar.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

systemctl uses the return codes defined by LSB, as defined in [LSB 3.0.0^{\[1\]}](#).

Table 3. LSB return codes

Value	Description in LSB	Use in systemd
0	"program is running or service is OK"	unit is active
1	"program is dead and /var/run pid file exists"	unit <i>not</i> failed (used by is-failed)
2	"program is dead and /var/lock lock file exists"	unused
3	"program is not running"	unit is not active
4	"program or service status is unknown"	no such unit

The mapping of LSB service states to systemd unit states is imperfect, so it is better to not rely on those return values but to look for specific unit states and substates instead.

ENVIRONMENT**\$SYSTEMD_EDITOR**

Editor to use when editing units; overrides **\$EDITOR** and **\$VISUAL**. If neither **\$SYSTEMD_EDITOR** nor **\$EDITOR** nor **\$VISUAL** are present or if it is set to an empty string or if their execution failed, **systemctl** will try to execute well known editors in this order: **editor(1)**, **nano(1)**, **vim(1)**, **vi(1)**.

\$SYSTEMD_LOG_LEVEL

The maximum log level of emitted messages (messages with a higher log level, i.e. less important

ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog(3)** for more information.

\$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This setting is only useful when messages are written directly to the terminal, because **journctl(1)** and other tools that display logs will color messages based on the log level on their own.

\$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This setting is only useful when messages are written directly to the terminal or a file, because **journctl(1)** and other tools that display logs will attach timestamps based on the entry metadata on their own.

\$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

Note that the log location is often attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console-prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see **syslog(3)**), **kmsg** (log to the kernel circular log buffer), **journal** (log to the journal), **journal-or-kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

\$SYSTEMD_PAGER

Pager to use when **--no-pager** is not given; overrides **\$PAGER**. If neither **\$SYSTEMD_PAGER** nor **\$PAGER** are set, a set of well-known pager implementations are tried in turn, including **less(1)** and **more(1)**, until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **--no-pager**.

\$SYSTEMD_LESS

Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K

This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of **\$SYSTEMD_LESS** does not include "K", and the pager that is invoked is **less**, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X

This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See **less(1)** for more discussion.

\$SYSTEMD_LESSCHARSET

Override the charset passed to **less** (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

\$SYSTEMD_PAGERSECURE

Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If **\$SYSTEMD_PAGERSECURE** is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see **geteuid(2)** and **sd_pid_get_owner_uid(3)**. In secure mode, **LESSSECURE=1** will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When **\$SYSTEMD_PAGERSECURE** is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only **less(1)** implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under **sudo(8)** or **pkexec(1)**, care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as described above. Setting **SYSTEMD_PAGERSECURE=0** or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the **\$SYSTEMD_PAGER** or **\$PAGER** variables are to be honoured, **\$SYSTEMD_PAGERSECURE** must be set too. It might be reasonable to completely disable the pager using **--no-pager** instead.

\$SYSTEMD_COLORS

Takes a boolean argument. When true, **systemd** and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on **\$TERM** and what the console is connected to.

\$SYSTEMD_URLIFY

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that **systemd** makes based on **\$TERM** and other conditions.

SEE ALSO

systemd(1), **journald(1)**, **logind(1)**, **machinectl(1)**, **systemd.unit(5)**, **systemd.resource-control(5)**, **systemd.special(7)**, **wall(1)**, **systemd.preset(5)**, **systemd.generator(7)**, **glob(7)**

NOTES

1. LSB 3.0.0
http://refspecs.linuxbase.org/LSB_3.0.0/LSB-PDA/LSB-PDA/iniscriptact.html

NAME

systemd-analyze – Analyze and debug system manager

SYNOPSIS

```
systemd-analyze [OPTIONS...] [time]
systemd-analyze [OPTIONS...] blame
systemd-analyze [OPTIONS...] critical-chain [UNIT...]
systemd-analyze [OPTIONS...] dump
systemd-analyze [OPTIONS...] plot [>file.svg]
systemd-analyze [OPTIONS...] dot [PATTERN...] [>file.dot]
systemd-analyze [OPTIONS...] unit-paths
systemd-analyze [OPTIONS...] exit-status [STATUS...]
systemd-analyze [OPTIONS...] capability [CAPABILITY...]
systemd-analyze [OPTIONS...] condition CONDITION...
systemd-analyze [OPTIONS...] syscall-filter [SET...]
systemd-analyze [OPTIONS...] calendar SPEC...
systemd-analyze [OPTIONS...] timestamp TIMESTAMP...
systemd-analyze [OPTIONS...] timespan SPAN...
systemd-analyze [OPTIONS...] cat-config NAME|PATH...
systemd-analyze [OPTIONS...] verify [FILE...]
systemd-analyze [OPTIONS...] security UNIT...
```

DESCRIPTION

systemd-analyze may be used to determine system boot-up performance statistics and retrieve other state and tracing information from the system and service manager, and to verify the correctness of unit files. It is also used to access special functions useful for advanced system manager debugging.

If no command is passed, **systemd-analyze time** is implied.

systemd-analyze time

This command prints the time spent in the kernel before userspace has been reached, the time spent in the initial RAM disk (initrd) before normal system userspace has been reached, and the time normal system userspace took to initialize. Note that these measurements simply measure the time passed up to the point where all system services have been spawned, but not necessarily until they fully finished initialization or the disk is idle.

Example 1. Show how long the boot took

```
# in a container
$ systemd-analyze time
Startup finished in 296ms (userspace)
multi-user.target reached after 275ms in userspace

# on a real machine
$ systemd-analyze time
Startup finished in 2.584s (kernel) + 19.176s (initrd) + 47.847s (userspace) = 1min 9.608s
multi-user.target reached after 47.820s in userspace
```

systemd-analyze blame

This command prints a list of all running units, ordered by the time they took to initialize. This information may be used to optimize boot-up times. Note that the output might be misleading as the initialization of one service might be slow simply because it waits for the initialization of another service to complete. Also

note: **systemd-analyze blame** doesn't display results for services with *Type=simple*, because systemd considers such services to be started immediately, hence no measurement of the initialization delays can be done. Also note that this command only shows the time units took for starting up, it does not show how long unit jobs spent in the execution queue. In particular it shows the time units spent in "activating" state, which is not defined for units such as device units that transition directly from "inactive" to "active". This command hence gives an impression of the performance of program code, but cannot accurately reflect latency introduced by waiting for hardware and similar events.

Example 2. Show which units took the most time during boot

```
$ systemd-analyze blame
 32.875s pmlogger.service
 20.905s systemd-networkd-wait-online.service
 13.299s dev-vda1.device
 ...
 23ms sysroot.mount
 11ms initrd-udevadm-cleanup-db.service
 3ms sys-kernel-config.mount
```

systemd-analyze critical-chain [UNIT...]

This command prints a tree of the time-critical chain of units (for each of the specified *UNITS* or for the default target otherwise). The time after the unit is active or started is printed after the "@" character. The time the unit takes to start is printed after the "+" character. Note that the output might be misleading as the initialization of services might depend on socket activation and because of the parallel execution of units. Also, similar to the **blame** command, this only takes into account the time units spent in "activating" state, and hence does not cover units that never went through an "activating" state (such as device units that transition directly from "inactive" to "active"). Moreover it does not show information on jobs (and in particular not jobs that timed out).

Example 3. systemd-analyze critical-chain

```
$ systemd-analyze critical-chain
multi-user.target @47.820s
pmie.service @35.968s +548ms
pmcd.service @33.715s +2.247s
network-online.target @33.712s
  systemd-networkd-wait-online.service @12.804s +20.905s
    systemd-networkd.service @11.109s +1.690s
      systemd-udevd.service @9.201s +1.904s
        systemd-tmpfiles-setup-dev.service @7.306s +1.776s
          kmod-static-nodes.service @6.976s +177ms
            systemd-journald.socket
              system.slice
                -.slice
```

systemd-analyze dump

This command outputs a (usually very long) human-readable serialization of the complete server state. Its format is subject to change without notice and should not be parsed by applications.

Example 4. Show the internal state of user manager

```
$ systemd-analyze --user dump
Timestamp userspace: Thu 2019-03-14 23:28:07 CET
Timestamp finish: Thu 2019-03-14 23:28:07 CET
Timestamp generators-start: Thu 2019-03-14 23:28:07 CET
Timestamp generators-finish: Thu 2019-03-14 23:28:07 CET
```

```

Timestamp units-load-start: Thu 2019-03-14 23:28:07 CET
Timestamp units-load-finish: Thu 2019-03-14 23:28:07 CET
-> Unit proc-timer_list.mount:
    Description: /proc/timer_list
    ...
-> Unit default.target:
    Description: Main user target
    ...

```

systemd-analyze plot

This command prints an SVG graphic detailing which system services have been started at what time, highlighting the time they spent on initialization.

Example 5. Plot a bootchart

```
$ systemd-analyze plot >bootup.svg
$ eog bootup.svg&
```

systemd-analyze dot [pattern...]

This command generates textual dependency graph description in dot format for further processing with the GraphViz **dot**(1) tool. Use a command line like **systemd-analyze dot | dot -Tsvg >systemd.svg** to generate a graphical dependency tree. Unless **--order** or **--require** is passed, the generated graph will show both ordering and requirement dependencies. Optional pattern globbing style specifications (e.g. `*.target`) may be given at the end. A unit dependency is included in the graph if any of these patterns match either the origin or destination node.

Example 6. Plot all dependencies of any unit whose name starts with "avahi-daemon"

```
$ systemd-analyze dot 'avahi-daemon.*' | dot -Tsvg >avahi.svg
$ eog avahi.svg
```

Example 7. Plot the dependencies between all known target units

```
$ systemd-analyze dot --to-pattern='*.target' --from-pattern='*.target' \
| dot -Tsvg >targets.svg
$ eog targets.svg
```

systemd-analyze unit-paths

This command outputs a list of all directories from which unit files, .d overrides, and .wants, .requires symlinks may be loaded. Combine with **--user** to retrieve the list for the user manager instance, and **--global** for the global configuration of user manager instances.

Example 8. Show all paths for generated units

```
$ systemd-analyze unit-paths | grep '^/run'
/run/systemd/system.control
/run/systemd/transient
/run/systemd/generator.early
/run/systemd/system
/run/systemd/system.attached
/run/systemd/generator
/run/systemd/generator.late
```

Note that this verb prints the list that is compiled into **systemd-analyze** itself, and does not communicate with the running manager. Use

```
systemctl [--user] [--global] show -p UnitPath --value
```

to retrieve the actual list that the manager uses, with any empty directories omitted.

systemd-analyze exit-status [STATUS...]

This command prints a list of exit statuses along with their "class", i.e. the source of the definition (one of "glibc", "systemd", "LSB", or "BSD"), see the Process Exit Codes section in **systemd.exec(5)**. If no additional arguments are specified, all known statuses are shown. Otherwise, only the definitions for the specified codes are shown.

Example 9. Show some example exit status names

```
$ systemd-analyze exit-status 0 1 {63..65}
NAME STATUS CLASS
SUCCESS 0 glibc
FAILURE 1 glibc
- 63 -
USAGE 64 BSD
DATAERR 65 BSD
```

systemd-analyze capability [CAPABILITY...]

This command prints a list of Linux capabilities along with their numeric IDs. See **capabilities(7)** for details. If no argument is specified the full list of capabilities known to the service manager and the kernel is shown. Capabilities defined by the kernel but not known to the service manager are shown as "cap_??". Optionally, if arguments are specified they may refer to specific capabilities by name or numeric ID, in which case only the indicated capabilities are shown in the table.

Example 10. Show some example capability names

```
$ systemd-analyze capability 0 1 {30..32}
NAME NUMBER
cap_chown 0
cap_dac_override 1
cap_audit_control 30
cap_setfcap 31
cap_mac_override 32
```

systemd-analyze condition CONDITION...

This command will evaluate *Condition*=...* and *Assert*=...* assignments, and print their values, and the resulting value of the combined condition set. See **systemd.unit(5)** for a list of available conditions and asserts.

Example 11. Evaluate conditions that check kernel versions

```
$ systemd-analyze condition 'ConditionKernelVersion = !<4.0' \
  'ConditionKernelVersion = >=5.1' \
  'ConditionACPower=false' \
  'ConditionArchitecture=!arm' \
  'AssertPathExists=/etc/os-release'
test.service: AssertPathExists=/etc/os-release succeeded.
Asserts succeeded.
test.service: ConditionArchitecture=!arm succeeded.
test.service: ConditionACPower=false failed.
test.service: ConditionKernelVersion=>=5.1 succeeded.
test.service: ConditionKernelVersion!=<4.0 succeeded.
Conditions succeeded.
```

systemd-analyze syscall-filter [SET...]

This command will list system calls contained in the specified system call set *SET*, or all known sets if no sets are specified. Argument *SET* must include the "@" prefix.

systemd-analyze calendar EXPRESSION...

This command will parse and normalize repetitive calendar time events, and will calculate when they elapse next. This takes the same input as the *OnCalendar=* setting in **systemd.timer(5)**, following the syntax described in **systemd.time(7)**. By default, only the next time the calendar expression will elapse is shown; use **--iterations=** to show the specified number of next times the expression elapses. Each time the expression elapses forms a timestamp, see the **timestamp** verb below.

Example 12. Show leap days in the near future

```
$ systemd-analyze calendar --iterations=5 '*-2-29 0:0:0'
Original form: *-2-29 0:0:0
Normalized form: *-02-29 00:00:00
Next elapse: Sat 2020-02-29 00:00:00 UTC
From now: 11 months 15 days left
Iter. #2: Thu 2024-02-29 00:00:00 UTC
From now: 4 years 11 months left
Iter. #3: Tue 2028-02-29 00:00:00 UTC
From now: 8 years 11 months left
Iter. #4: Sun 2032-02-29 00:00:00 UTC
From now: 12 years 11 months left
Iter. #5: Fri 2036-02-29 00:00:00 UTC
From now: 16 years 11 months left
```

systemd-analyze timestamp TIMESTAMP...

This command parses a timestamp (i.e. a single point in time) and outputs the normalized form and the difference between this timestamp and now. The timestamp should adhere to the syntax documented in **systemd.time(7)**, section "PARSING TIMESTAMPS".

Example 13. Show parsing of timestamps

```
$ systemd-analyze timestamp yesterday now tomorrow
Original form: yesterday
Normalized form: Mon 2019-05-20 00:00:00 CEST
(in UTC): Sun 2019-05-19 22:00:00 UTC
UNIX seconds: @15583032000
From now: 1 day 9h ago

Original form: now
Normalized form: Tue 2019-05-21 09:48:39 CEST
(in UTC): Tue 2019-05-21 07:48:39 UTC
UNIX seconds: @1558424919.659757
From now: 43us ago
```

```
Original form: tomorrow
Normalized form: Wed 2019-05-22 00:00:00 CEST
(in UTC): Tue 2019-05-21 22:00:00 UTC
UNIX seconds: @15584760000
From now: 14h left
```

systemd-analyze timespan EXPRESSION...

This command parses a time span (i.e. a difference between two timestamps) and outputs the normalized form and the equivalent value in microseconds. The time span should adhere to the syntax documented in **systemd.time(7)**, section "PARSING TIME SPANS". Values without units are parsed as seconds.

Example 14. Show parsing of timespans

```
$ systemd-analyze timespan 1s 300s '1year 0.000001s'
```

```
Original: 1s
  μs: 1000000
Human: 1s
```

```
Original: 300s
  μs: 300000000
Human: 5min
```

```
Original: 1year 0.000001s
  μs: 315576000000001
Human: 1y 1us
```

systemd-analyze cat–config NAME|PATH...

This command is similar to **systemctl cat**, but operates on config files. It will copy the contents of a config file and any drop-ins to standard output, using the usual systemd set of directories and rules for precedence. Each argument must be either an absolute path including the prefix (such as /etc/systemd/logind.conf or /usr/lib/systemd/logind.conf), or a name relative to the prefix (such as systemd/logind.conf).

Example 15. Showing logind configuration

```
$ systemd-analyze cat–config systemd/logind.conf
# /etc/systemd/logind.conf
...
[Login]
NAutoVTs=8
...
# /usr/lib/systemd/logind.conf.d/20–test.conf
... some override from another package
# /etc/systemd/logind.conf.d/50–override.conf
... some administrator override
```

systemd-analyze verify FILE...

This command will load unit files and print warnings if any errors are detected. Files specified on the command line will be loaded, but also any other units referenced by them. The full unit search path is formed by combining the directories for all command line arguments, and the usual unit load paths. The variable \$SYSTEMD_UNIT_PATH is supported, and may be used to replace or augment the compiled in set of unit load paths; see **systemd.unit(5)**. All units files present in the directories containing the command line arguments will be used in preference to the other paths.

The following errors are currently detected:

- unknown sections and directives,
- missing dependencies which are required to start the given unit,
- man pages listed in *Documentation=* which are not found in the system,
- commands listed in *ExecStart=* and similar which are not found in the system or not executable.

Example 16. Misspelt directives

```
$ cat ./user.slice
[Unit]
WhatIsThis=11
Documentation=man:nosuchfile(1)
Requires=different.service
```

```
[Service]
Description=x

$ systemd-analyze verify ./user.slice
[./user.slice:9] Unknown lvalue 'WhatIsThis' in section 'Unit'
[./user.slice:13] Unknown section 'Service'. Ignoring.
Error: org.freedesktop.systemd1.LoadFailed:
  Unit different.service failed to load:
    No such file or directory.
Failed to create user.slice/start: Invalid argument
user.slice: man nosuchfile(1) command failed with code 16
```

Example 17. Missing service units

```
$ tail ./a.socket ./b.socket
==> ./a.socket <==
[Socket]
ListenStream=100

==> ./b.socket <==
[Socket]
ListenStream=100
Accept=yes

$ systemd-analyze verify ./a.socket ./b.socket
Service a.service not loaded, a.socket cannot be started.
Service b@0.service not loaded, b.socket cannot be started.
```

systemd-analyze security [UNIT...]

This command analyzes the security and sandboxing settings of one or more specified service units. If at least one unit name is specified the security settings of the specified service units are inspected and a detailed analysis is shown. If no unit name is specified, all currently loaded, long-running service units are inspected and a terse table with results shown. The command checks for various security-related service settings, assigning each a numeric "exposure level" value, depending on how important a setting is. It then calculates an overall exposure level for the whole unit, which is an estimation in the range 0.0...10.0 indicating how exposed a service is security-wise. High exposure levels indicate very little applied sandboxing. Low exposure levels indicate tight sandboxing and strongest security restrictions. Note that this only analyzes the per-service security features systemd itself implements. This means that any additional security mechanisms applied by the service code itself are not accounted for. The exposure level determined this way should not be misunderstood: a high exposure level neither means that there is no effective sandboxing applied by the service code itself, nor that the service is actually vulnerable to remote or local attacks. High exposure levels do indicate however that most likely the service might benefit from additional settings applied to them.

Please note that many of the security and sandboxing settings individually can be circumvented — unless combined with others. For example, if a service retains the privilege to establish or undo mount points many of the sandboxing options can be undone by the service code itself. Due to that it is essential that each service uses the most comprehensive and strict sandboxing and security settings possible. The tool will take into account some of these combinations and relationships between the settings, but not all. Also note that the security and sandboxing settings analyzed here only apply to the operations executed by the service code itself. If a service has access to an IPC system (such as D-Bus) it might request operations from other services that are not subject to the same restrictions. Any comprehensive security and sandboxing analysis is hence incomplete if the IPC access policy is not validated too.

Example 18. Analyze systemd-logind.service

```
$ systemd-analyze security --no-pager systemd-logind.service
  NAME          DESCRIPTION          EXPOSURE
PrivateNetwork=  Service has access to the host's network  0.5
User=/DynamicUser= Service runs as root user            0.4
DeviceAllow=    Service has no device ACL                0.2
✓ IPAddressDeny= Service blocks all IP address ranges
...
→ Overall exposure level for systemd-logind.service: 4.1 OK
```

OPTIONS

The following options are understood:

--system

Operates on the system systemd instance. This is the implied default.

--user

Operates on the user systemd instance.

--global

Operates on the system-wide configuration for user systemd instance.

--order, --require

When used in conjunction with the **dot** command (see above), selects which dependencies are shown in the dependency graph. If **--order** is passed, only dependencies of type *After*= or *Before*= are shown. If **--require** is passed, only dependencies of type *Requires*=, *Requisite*=, *Wants*= and *Conflicts*= are shown. If neither is passed, this shows dependencies of all these types.

--from-pattern=, --to-pattern=

When used in conjunction with the **dot** command (see above), this selects which relationships are shown in the dependency graph. Both options require a **glob(7)** pattern as an argument, which will be matched against the left-hand and the right-hand, respectively, nodes of a relationship.

Each of these can be used more than once, in which case the unit name must match one of the values. When tests for both sides of the relation are present, a relation must pass both tests to be shown. When patterns are also specified as positional arguments, they must match at least one side of the relation. In other words, patterns specified with those two options will trim the list of edges matched by the positional arguments, if any are given, and fully determine the list of edges shown otherwise.

--fuzz=*timespan*

When used in conjunction with the **critical-chain** command (see above), also show units, which finished *timespan* earlier, than the latest unit in the same level. The unit of *timespan* is seconds unless specified with a different unit, e.g. "50ms".

--man=no

Do not invoke **man(1)** to verify the existence of man pages listed in *Documentation*=.

--generators

Invoke unit generators, see **systemd.generator(7)**. Some generators require root privileges. Under a normal user, running with generators enabled will generally result in some warnings.

--root=PATH

With **cat-files**, show config files underneath the specified root path *PATH*.

--iterations=NUMBER

When used with the **calendar** command, show the specified number of iterations the specified calendar expression will elapse next. Defaults to 1.

--base-time=TIMESTAMP

When used with the **calendar** command, show next iterations relative to the specified point in time. If not specified defaults to the current time.

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":"; and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--no-pager

Do not pipe output into a pager.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

ENVIRONMENT***\$SYSTEMD_LOG_LEVEL***

The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog(3)** for more information.

\$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This setting is only useful when messages are written directly to the terminal, because **journalctl(1)** and other tools that display logs will color messages based on the log level on their own.

\$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This setting is only useful when messages are written directly to the terminal or a file, because **journalctl(1)** and other tools that display logs will attach timestamps based on the entry metadata on their own.

\$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

Note that the log location is often attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TID

A boolean. If true, messages will be prefixed with the current numerical thread ID (TID).

Note that the this information is attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console-prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see **syslog(3)**), **kmsg** (log to the

kernel circular log buffer), **journal** (log to the journal), **journal-or-kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

\$SYSTEMD_PAGER

Pager to use when **--no-pager** is not given; overrides **\$PAGER**. If neither **\$SYSTEMD_PAGER** nor **\$PAGER** are set, a set of well-known pager implementations are tried in turn, including **less(1)** and **more(1)**, until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **--no-pager**.

\$SYSTEMD_LESS

Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K

This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of **\$SYSTEMD_LESS** does not include "K", and the pager that is invoked is **less**, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X

This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See **less(1)** for more discussion.

\$SYSTEMD_LESSCHARSET

Override the charset passed to **less** (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

\$SYSTEMD_PAGERSECURE

Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If **\$SYSTEMD_PAGERSECURE** is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see **geteuid(2)** and **sd_pid_get_owner_uid(3)**. In secure mode, **LESSSECURE=1** will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When **\$SYSTEMD_PAGERSECURE** is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only **less(1)** implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under **sudo(8)** or **pkexec(1)**, care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as described above. Setting **SYSTEMD_PAGERSECURE=0** or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the **\$SYSTEMD_PAGER** or **\$PAGER** variables are to be honoured,

\$SYSTEMD_PAGERSECURE must be set too. It might be reasonable to completely disable the pager using **--no-pager** instead.

\$SYSTEMD_COLORS

Takes a boolean argument. When true, **systemd** and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on **\$TERM** and what the console is connected to.

\$SYSTEMD_URLIFY

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that **systemd** makes based on *\$TERM* and other conditions.

SEE ALSO

systemd(1), **systemctl(1)**

NAME

systemd-fsck@.service, systemd-fsck-root.service, systemd-fsck – File system checker logic

SYNOPSIS

```
systemd-fsck@.service
systemd-fsck-root.service
/lib/systemd/systemd-fsck
```

DESCRIPTION

systemd-fsck@.service and systemd-fsck-root.service are services responsible for file system checks. They are instantiated for each device that is configured for file system checking. systemd-fsck-root.service is responsible for file system checks on the root file system, but only if the root filesystem was not checked in the initramfs. systemd-fsck@.service is used for all other file systems and for the root file system in the initramfs.

These services are started at boot if **passno** in /etc/fstab for the file system is set to a value greater than zero. The file system check for root is performed before the other file systems. Other file systems may be checked in parallel, except when they are on the same rotating disk.

systemd-fsck does not know any details about specific filesystems, and simply executes file system checkers specific to each filesystem type (/sbin/fsck.type). These checkers will decide if the filesystem should actually be checked based on the time since last check, number of mounts, unclean unmount, etc.

If a file system check fails for a service without **nofail**, emergency mode is activated, by isolating to emergency.target.

KERNEL COMMAND LINE

systemd-fsck understands these kernel command line parameters:

fsck.mode=

One of "auto", "force", "skip". Controls the mode of operation. The default is "auto", and ensures that file system checks are done when the file system checker deems them necessary. "force" unconditionally results in full file system checks. "skip" skips any file system checks.

fsck.repair=

One of "preen", "yes", "no". Controls the mode of operation. The default is "preen", and will automatically repair problems that can be safely fixed. "yes" will answer yes to all questions by fsck and "no" will answer no to all questions.

SEE ALSO

systemd(1), fsck(8), systemd-quotacheck.service(8), fsck.btrfs(8), fsck.cramfs(8), fsck.ext4(8), fsck.fat(8), fsck.hfsplus(8), fsck.minix(8), fsck.ntfs(8), fsck.xfs(8)

NAME

systemd-hostnamed.service, systemd-hostnamed – Daemon to control system hostname from programs

SYNOPSIS

systemd-hostnamed.service

/lib/systemd/systemd-hostnamed

DESCRIPTION

systemd-hostnamed.service is a system service that may be used to change the system's hostname and related machine metadata from user programs. It is automatically activated on request and terminates itself when unused.

It currently offers access to five variables:

- The current hostname (Example: "dhcp-192-168-47-11")
- The static (configured) hostname (Example: "lennarts-computer")
- The pretty hostname (Example: "Lennart's Computer")
- A suitable icon name for the local host (Example: "computer-laptop")
- A chassis type (Example: "tablet")

The static hostname is stored in /etc/hostname, see **hostname(5)** for more information. The pretty hostname, chassis type, and icon name are stored in /etc/machine-info, see **machine-info(5)**.

The tool **hostnamectl(1)** is a command line client to this service.

See **org.freedesktop.hostname1(5)** and **org.freedesktop.LogControl1(5)** for a description of the D-Bus API.

SEE ALSO

systemd(1), **hostname(5)**, **machine-info(5)**, **hostnamectl(1)**, **sethostname(2)**

NAME

systemd-journald.service, systemd-journald.socket, systemd-journald-dev-log.socket, systemd-journald-audit.socket, systemd-journald@.service, systemd-journald@.socket, systemd-journald-varlink@.socket, systemd-journald – Journal service

SYNOPSIS

```
systemd-journald.service
systemd-journald.socket
systemd-journald-dev-log.socket
systemd-journald-audit.socket
systemd-journald@.service
systemd-journald@.socket
systemd-journald-varlink@.socket
/lib/systemd/systemd-journald
```

DESCRIPTION

systemd-journald is a system service that collects and stores logging data. It creates and maintains structured, indexed journals based on logging information that is received from a variety of sources:

- Kernel log messages, via kmsg
- Simple system log messages, via the libc **syslog(3)** call
- Structured system log messages via the native Journal API, see **sd_journal_print(3)** and **Native Journal Protocol**^[1]
- Standard output and standard error of service units. For further details see below.
- Audit records, originating from the kernel audit subsystem

The daemon will implicitly collect numerous metadata fields for each log messages in a secure and unforgeable way. See **systemd.journal-fields(7)** for more information about the collected metadata.

Log data collected by the journal is primarily text-based but can also include binary data where necessary. Individual fields making up a log record stored in the journal may be up to $2^{64}-1$ bytes in size.

The journal service stores log data either persistently below `/var/log/journal` or in a volatile way below `/run/log/journal/` (in the latter case it is lost at reboot). By default, log data is stored persistently if `/var/log/journal/` exists during boot, with an implicit fallback to volatile storage otherwise. Use `Storage=` in **journald.conf(5)** to configure where log data is placed, independently of the existence of `/var/log/journal/`.

Note that journald will initially use volatile storage, until a call to **journalctl --flush** (or sending **SIGUSR1** to journald) will cause it to switch to persistent logging (under the conditions mentioned above). This is done automatically on boot via "systemd-journal-flush.service".

On systems where `/var/log/journal/` does not exist yet but where persistent logging is desired (and the default **journald.conf** is used), it is sufficient to create the directory, and ensure it has the correct access modes and ownership:

```
mkdir -p /var/log/journal
systemd-tmpfiles --create --prefix /var/log/journal
```

See **journald.conf(5)** for information about the configuration of this service.

STREAM LOGGING

The systemd service manager invokes all service processes with standard output and standard error connected to the journal by default. This behaviour may be altered via the `StandardOutput=/StandardError=` unit file settings, see **systemd.exec(5)** for details. The journal converts the log byte stream received this way into individual log records, splitting the stream at newline ("\\n", ASCII **10**) and **NUL** bytes.

If `systemd-journald.service` is stopped, the stream connections associated with all services are terminated. Further writes to those streams by the service will result in **EPIPE** errors. In order to react gracefully in this case it is recommended that programs logging to standard output/error ignore such errors. If the **SIGPIPE** UNIX signal handler is not blocked or turned off, such write attempts will also result in such process signals being generated, see `signal(7)`. To mitigate this issue, systemd service manager explicitly turns off the **SIGPIPE** signal for all invoked processes by default (this may be changed for each unit individually via the `IgnoreSIGPIPE=` option, see `systemd.exec(5)` for details). After the standard output/standard error streams have been terminated they may not be recovered until the services they are associated with are restarted. Note that during normal operation, `systemd-journald.service` stores copies of the file descriptors for those streams in the service manager. If `systemd-journald.service` is restarted using `systemctl restart` or equivalent operation instead of a pair of separate `systemctl stop` and `systemctl start` commands (or equivalent operations), these stream connections are not terminated and survive the restart. It is thus safe to restart `systemd-journald.service`, but stopping it is not recommended.

Note that the log record metadata for records transferred via such standard output/error streams reflect the metadata of the peer the stream was originally created for. If the stream connection is passed on to other processes (such as further child processes forked off the main service process), the log records will not reflect their metadata, but will continue to describe the original process. This is different from the other logging transports listed above, which are inherently record based and where the metadata is always associated with the individual record.

In addition to the implicit standard output/error logging of services, stream logging is also available via the `systemd-cat(1)` command line tool.

Currently, the number of parallel log streams `systemd-journald` will accept is limited to 4096. When this limit is reached further log streams may be established but will receive **EPIPE** right from the beginning.

JOURNAL NAMESPACES

Journal 'namespaces' are both a mechanism for logically isolating the log stream of projects consisting of one or more services from the rest of the system and a mechanism for improving performance. Multiple journal namespaces may exist simultaneously, each defining its own, independent log stream managed by its own instance of `systemd-journald`. Namespaces are independent of each other, both in the data store and in the IPC interface. By default only a single 'default' namespace exists, managed by `systemd-journald.service` (and its associated socket units). Additional namespaces are created by starting an instance of the `systemd-journald@.service` service template. The instance name is the namespace identifier, which is a short string used for referencing the journal namespace. Service units may be assigned to a specific journal namespace through the `LogNamespace=` unit file setting, see `systemd.exec(5)` for details. The `--namespace=` switch of `journalctl(1)` may be used to view the log stream of a specific namespace. If the switch is not used the log stream of the default namespace is shown, i.e. log data from other namespaces is not visible.

Services associated with a specific log namespace may log via syslog, the native logging protocol of the journal and via `stdout/stderr`; the logging from all three transports is associated with the namespace.

By default only the default namespace will collect kernel and audit log messages.

The `systemd-journald` instance of the default namespace is configured through `/etc/systemd/journald.conf` (see below), while the other instances are configured through `/etc/systemd/journald@NAMESPACE.conf`. The journal log data for the default namespace is placed in `/var/log/journal/MACHINE_ID` (see below) while the data for the other namespaces is located in `/var/log/journal/MACHINE_ID.NAMESPACE`.

SIGNALS

SIGUSR1

Request that journal data from `/run/` is flushed to `/var/` in order to make it persistent (if this is enabled). This must be used after `/var/` is mounted, as otherwise log data from `/run/` is never flushed to `/var/` regardless of the configuration. Use the `journalctl --flush` command to request flushing of the journal files, and wait for the operation to complete. See `journalctl(1)` for details.

SIGUSR2

Request immediate rotation of the journal files. Use the `journalctl --rotate` command to request

journal file rotation, and wait for the operation to complete.

SIGRTMIN+1

Request that all unwritten log data is written to disk. Use the **journaldctl --sync** command to trigger journal synchronization, and wait for the operation to complete.

KERNEL COMMAND LINE

A few configuration parameters from journald.conf may be overridden on the kernel command line:

systemd.journald.forward_to_syslog=, *systemd.journald.forward_to_kmsg=*,
systemd.journald.forward_to_console=, *systemd.journald.forward_to_wall=*

Enables/disables forwarding of collected log messages to syslog, the kernel log buffer, the system console or wall.

See **journald.conf(5)** for information about these settings.

Note that these kernel command line options are only honoured by the default namespace, see above.

ACCESS CONTROL

Journal files are, by default, owned and readable by the "systemd-journal" system group but are not writable. Adding a user to this group thus enables them to read the journal files.

By default, each user, with a UID outside the range of system users, dynamic service users, and the nobody user, will get their own set of journal files in /var/log/journal/. See **Users, Groups, UIDs and GIDs on systemd systems**^[2] for more details about UID ranges. These journal files will not be owned by the user, however, in order to avoid that the user can write to them directly. Instead, file system ACLs are used to ensure the user gets read access only.

Additional users and groups may be granted access to journal files via file system access control lists (ACL). Distributions and administrators may choose to grant read access to all members of the "wheel" and "adm" system groups with a command such as the following:

```
# setfacl -Rnm g:wheel:rx,d:g:wheel:rx,g:adm:rx,d:g:adm:rx /var/log/journal/
```

Note that this command will update the ACLs both for existing journal files and for future journal files created in the /var/log/journal/ directory.

FILES

/etc/systemd/journald.conf

Configure **systemd-journald** behavior. See **journald.conf(5)**.

/run/log/journal/*machine-id*/*.journal, /run/log/journal/*machine-id*/*.journal~,
/var/log/journal/*machine-id*/*.journal, /var/log/journal/*machine-id*/*.journal~

systemd-journald writes entries to files in /run/log/journal/*machine-id*/ or
/var/log/journal/*machine-id*/ with the ".journal" suffix. If the daemon is stopped uncleanly, or if the files are found to be corrupted, they are renamed using the ".journal~" suffix, and **systemd-journald** starts writing to a new file. /run/ is used when /var/log/journal is not available, or when **Storage=volatile** is set in the **journald.conf(5)** configuration file.

When **systemd-journald** ceases writing to a journal file, it will be renamed to "*original-name@suffix.journal*" (or "*original-name@suffix.journal~*"). Such files are "archived" and will not be written to any more.

In general, it is safe to read or copy any journal file (active or archived). **journaldctl(1)** and the functions in the **sd-journal(3)** library should be able to read all entries that have been fully written.

systemd-journald will automatically remove the oldest archived journal files to limit disk use. See **SystemMaxUse=** and related settings in **journald.conf(5)**.

/dev/kmsg, /dev/log, /run/systemd/journal/dev-log, /run/systemd/journal/socket,
/run/systemd/journal/stdout

Sockets and other file node paths that **systemd-journald** will listen on and are visible in the file system. In addition to these, **systemd-journald** can listen for audit events using **netlink(7)**.

If journal namespacing is used these paths are slightly altered to include a namespace identifier, see above.

SEE ALSO

systemd(1), journalctl(1), journald.conf(5), systemd.journal-fields(7), sd-journal(3), systemd-coredump(8), setfacl(1), sd_journal_print(3), pydoc systemd.journal

NOTES

1. Native Journal Protocol
https://systemd.io/JOURNAL_NATIVE_PROTOCOL
2. Users, Groups, UIDs and GIDs on systemd systems
<https://systemd.io/UIDS-GIDS>

NAME

`systemd-machined.service`, `systemd-machined` – Virtual machine and container registration manager

SYNOPSIS

```
systemd-machined.service
/lib/systemd/systemd-machined
```

DESCRIPTION

systemd-machined is a system service that keeps track of locally running virtual machines and containers.

systemd-machined is useful for registering and keeping track of both OS containers (containers that share the host kernel but run a full init system of their own and behave in most regards like a full virtual operating system rather than just one virtualized app) and full virtual machines (virtualized hardware running normal operating systems and possibly different kernels).

systemd-machined should *not* be used for registering/keeping track of application sandbox containers. A *machine* in the context of **systemd-machined** is supposed to be an abstract term covering both OS containers and full virtual machines, but not application sandboxes.

Machines registered with machined are exposed in various ways in the system. For example:

- Tools like **ps**(1) will show to which machine a specific process belongs in a column of its own, and so will **gnome-system-monitor**^[1] or **systemd-cgls**(1).
- `systemctl`'s various tools (**systemctl**(1), **journalctl**(1), **logindctl**(1), **hostnamectl**(1), **timedatectl**(1), **localectl**(1), **machinectl**(1), ...) support the **-M** switch to operate on local containers instead of the host system.
- **systemctl list-machines** will show the system state of all local containers, connecting to the container's init system for that.
- `systemctl`'s **--recursive** switch has the effect of not only showing the locally running services, but recursively showing the services of all registered containers.
- The **machinectl** command provides access to a number of useful operations on registered containers, such as introspecting them, rebooting, shutting them down, and getting a login prompt on them.
- The **sd-bus**(3) library exposes the **sd_bus_open_system_machine**(3) call to connect to the system bus of any registered container.
- The **nss-myrmachines**(8) module makes sure all registered containers can be resolved via normal glibc **gethostbyname**(3) or **getaddrinfo**(3) calls.

See **systemd-nspawn**(1) for some examples on how to run containers with OS tools.

If you are interested in writing a VM or container manager that makes use of machined, please have look at [Writing Virtual Machine or Container Managers](#)^[2]. Also see the [New Control Group Interfaces](#)^[3].

The daemon provides both a C library interface (which is shared with **systemd-logind.service**(8)) as well as a D-Bus interface. The library interface may be used to introspect and watch the state of virtual machines/containers. The bus interface provides the same but in addition may also be used to register or terminate machines. For more information please consult **sd-login**(3) and **org.freedesktop.machine1**(5) and **org.freedesktop.LogControl1**(5).

A small companion daemon **systemd-importd.service**(8) is also available, which implements importing, exporting, and downloading of container and VM images.

For each container registered with `systemd-machined.service` that employs user namespacing, users/groups are synthesized for the used UIDs/GIDs. These are made available to the system using the [User/Group Record Lookup API via Varlink](#)^[4], and thus may be resolved with **userdbctl**(1) or the usual glibc NSS calls.

SEE ALSO

systemd(1), machinectl(1), systemd-nspawn(1), nss-myrmachines(8), systemd.special(7)

NOTES

1. gnome-system-monitor
<https://help.gnome.org/users/gnome-system-monitor/>
2. Writing Virtual Machine or Container Managers
<https://www.freedesktop.org/wiki/Software/systemd/writing-vm-managers>
3. New Control Group Interfaces
<https://www.freedesktop.org/wiki/Software/systemd/ControlGroupInterface/>
4. User/Group Record Lookup API via Varlink
https://systemd.io/USER_GROUP_API

NAME

systemd-mount, **systemd-umount** – Establish and destroy transient mount or auto-mount points

SYNOPSIS

```
systemd-mount [OPTIONS...] WHAT [WHERE]
systemd-mount [OPTIONS...] --list
systemd-mount [OPTIONS...] --umount WHAT/WHERE...
```

DESCRIPTION

systemd-mount may be used to create and start a transient .mount or .automount unit of the file system *WHAT* on the mount point *WHERE*.

In many ways, **systemd-mount** is similar to the lower-level **mount**(8) command, however instead of executing the mount operation directly and immediately, **systemd-mount** schedules it through the service manager job queue, so that it may pull in further dependencies (such as parent mounts, or a file system checker to execute a priori), and may make use of the auto-mounting logic.

The command takes either one or two arguments. If only one argument is specified it should refer to a block device or regular file containing a file system (e.g. "/dev/sdb1" or "/path/to/disk.img"). The block device or image file is then probed for a file system label and other metadata, and is mounted to a directory below /run/media/system/ whose name is generated from the file system label. In this mode the block device or image file must exist at the time of invocation of the command, so that it may be probed. If the device is found to be a removable block device (e.g. a USB stick), an automount point is created instead of a regular mount point (i.e. the **--automount**= option is implied, see below).

If two arguments are specified, the first indicates the mount source (the *WHAT*) and the second indicates the path to mount it on (the *WHERE*). In this mode no probing of the source is attempted, and a backing device node doesn't have to exist. However, if this mode is combined with **--discover**, device node probing for additional metadata is enabled, and – much like in the single-argument case discussed above – the specified device has to exist at the time of invocation of the command.

Use the **--list** command to show a terse table of all local, known block devices with file systems that may be mounted with this command.

systemd-umount can be used to unmount a mount or automount point. It is the same as **systemd-mount** **--umount**.

OPTIONS

The following options are understood:

--no-block

Do not synchronously wait for the requested operation to finish. If this is not specified, the job will be verified, enqueued and **systemd-mount** will wait until the mount or automount unit's start-up is completed. By passing this argument, it is only verified and enqueued.

-l, --full

Do not ellipsize the output when **--list** is specified.

--no-pager

Do not pipe output into a pager.

--no-legend

Do not print the legend, i.e. column headers and the footer with hints.

--no-ask-password

Do not query the user for authentication for privileged operations.

--quiet, -q

Suppresses additional informational output while running.

--discover

Enable probing of the mount source. This switch is implied if a single argument is specified on the command line. If passed, additional metadata is read from the device to enhance the unit to create. For

example, a descriptive string for the transient units is generated from the file system label and device model. Moreover if a removable block device (e.g. USB stick) is detected an automount unit instead of a regular mount unit is created, with a short idle timeout, in order to ensure the file-system is placed in a clean state quickly after each access.

--type=, -t

Specifies the file system type to mount (e.g. "vfat" or "ext4"). If omitted or set to "auto", the file system type is determined automatically.

--options=, -o

Additional mount options for the mount point.

--owner=USER

Let the specified user *USER* own the mounted file system. This is done by appending **uid=** and **gid=** options to the list of mount options. Only certain file systems support this option.

--fsck=

Takes a boolean argument, defaults to on. Controls whether to run a file system check immediately before the mount operation. In the automount case (see **--automount** below) the check will be run the moment the first access to the device is made, which might slightly delay the access.

--description=

Provide a description for the mount or automount unit. See *Description=* in **systemd.unit(5)**.

--property=, -p

Sets a unit property for the mount unit that is created. This takes an assignment in the same format as **systemctl(1)**'s **set-property** command.

--automount=

Takes a boolean argument. Controls whether to create an automount point or a regular mount point. If true an automount point is created that is backed by the actual file system at the time of first access. If false a plain mount point is created that is backed by the actual file system immediately. Automount points have the benefit that the file system stays unmounted and hence in clean state until it is first accessed. In automount mode the **--timeout-idle-sec=** switch (see below) may be used to ensure the mount point is unmounted automatically after the last access and an idle period passed.

If this switch is not specified it defaults to false. If not specified and **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, it is set to true, in order to increase the chance that the file system is in a fully clean state if the device is unplugged abruptly.

-A

Equivalent to **--automount=yes**.

--timeout-idle-sec=

Takes a time value that controls the idle timeout in automount mode. If set to "infinity" (the default) no automatic unmounts are done. Otherwise the file system backing the automount point is detached after the last access and the idle timeout passed. See **systemd.time(7)** for details on the time syntax supported. This option has no effect if only a regular mount is established, and automounting is not used.

Note that if **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, **--timeout-idle-sec=1s** is implied.

--automount-property=

Similar to **--property=**, but applies additional properties to the automount unit created, instead of the mount unit.

--bind-device

This option only has an effect in automount mode, and controls whether the automount unit shall be

bound to the backing device's lifetime. If set, the automount point will be removed automatically when the backing device vanishes. By default the automount point stays around, and subsequent accesses will block until backing device is replugged. This option has no effect in case of non-device mounts, such as network or virtual file system mounts.

Note that if **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, this option is implied.

--list

Instead of establishing a mount or automount point, print a terse list of block devices containing file systems that may be mounted with "systemd-mount", along with useful metadata such as labels, etc.

-u, --umount

Stop the mount and automount units corresponding to the specified mount points *WHERE* or the devices *WHAT*. **systemd-mount** with this option or **systemd-umount** can take multiple arguments which can be mount points, devices, /etc/fstab style node names, or backing files corresponding to loop devices, like **systemd-mount --umount /path/to/umount /dev/sda1 UUID=xxxxxx-xxxx LABEL=xxxxx /path/to/disk.img**. Note that when **-H** or **-M** is specified, only absolute paths to mount points are supported.

-G, --collect

Unload the transient unit after it completed, even if it failed. Normally, without this option, all mount units that mount and failed are kept in memory until the user explicitly resets their failure state with **systemctl reset-failed** or an equivalent command. On the other hand, units that stopped successfully are unloaded immediately. If this option is turned on the "garbage collection" of units is more aggressive, and unloads units regardless if they exited successfully or failed. This option is a shortcut for **--property=CollectMode=inactive-or-failed**, see the explanation for *CollectMode*= in **systemd.unit(5)** for further information.

--user

Talk to the service manager of the calling user, rather than the service manager of the system.

--system

Talk to the service manager of the system. This is the implied default.

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":"; and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

THE UDEV DATABASE

If **--discover** is used, **systemd-mount** honors a couple of additional udev properties of block devices:

SYSTEMD_MOUNT_OPTIONS=

The mount options to use, if **--options=** is not used.

SYSTEMD_MOUNT_WHERE=

The file system path to place the mount point at, instead of the automatically generated one.

EXAMPLE

Use a udev rule like the following to automatically mount all USB storage plugged in:

```
ACTION=="add", SUBSYSTEMS=="usb", SUBSYSTEM=="block", ENV{ID_FS_USAGE}=="filesystem", \
    RUN{program}+="/usr/bin/systemd-mount --no-block --automount=yes --collect $devnode"
```

SEE ALSO

systemd(1), mount(8), systemctl(1), systemd.unit(5), systemd.mount(5), systemd.automount(5),
systemd-run(1)

NAME

systemd-networkd.service, systemd-networkd – Network manager

SYNOPSIS

systemd–networkd.service

/lib/systemd/systemd–networkd

DESCRIPTION

systemd–networkd is a system service that manages networks. It detects and configures network devices as they appear, as well as creating virtual network devices.

To configure low-level link settings independently of networks, see **systemd.link(5)**.

systemd–networkd will create network devices based on the configuration in **systemd.netdev(5)** files, respecting the [Match] sections in those files.

systemd–networkd will manage network addresses and routes for any link for which it finds a .network file with an appropriate [Match] section, see **systemd.network(5)**. For those links, it will flush existing network addresses and routes when bringing up the device. Any links not matched by one of the .network files will be ignored. It is also possible to explicitly tell systemd–networkd to ignore a link by using *Unmanaged=yes* option, see **systemd.network(5)**.

When systemd–networkd exits, it generally leaves existing network devices and configuration intact. This makes it possible to transition from the initramfs and to restart the service without breaking connectivity. This also means that when configuration is updated and systemd–networkd is restarted, netdev interfaces for which configuration was removed will not be dropped, and may need to be cleaned up manually.

systemd–networkd may be introspected and controlled at runtime using **networkctl(1)**.

CONFIGURATION FILES

The configuration files are read from the files located in the system network directory /lib/systemd/network, the volatile runtime network directory /run/systemd/network and the local administration network directory /etc/systemd/network.

Networks are configured in .network files, see **systemd.network(5)**, and virtual network devices are configured in .netdev files, see **systemd.netdev(5)**.

SEE ALSO

networkctl(1), **systemd(1)**, **systemd.link(5)**, **systemd.network(5)**, **systemd.netdev(5)**, **systemd-networkd-wait-online.service(8)**, **systemd-network-generator.service(8)**

NAME

systemd-resolved.service, systemd-resolved – Network Name Resolution manager

SYNOPSIS

systemd-resolved.service

/lib/systemd/systemd-resolved

DESCRIPTION

systemd-resolved is a system service that provides network name resolution to local applications. It implements a caching and validating DNS/DNSSEC stub resolver, as well as an LLMNR and MulticastDNS resolver and responder. Local applications may submit network name resolution requests via three interfaces:

- The native, fully-featured API **systemd-resolved** exposes on the bus, see **org.freedesktop.resolve1(5)** and **org.freedesktop.LogControl1(5)** for details. Usage of this API is generally recommended to clients as it is asynchronous and fully featured (for example, properly returns DNSSEC validation status and interface scope for addresses as necessary for supporting link-local networking).
- The glibc **getaddrinfo(3)** API as defined by [RFC3493^{\[1\]}](#) and its related resolver functions, including **gethostbyname(3)**. This API is widely supported, including beyond the Linux platform. In its current form it does not expose DNSSEC validation status information however, and is synchronous only. This API is backed by the glibc Name Service Switch (**nss(5)**). Usage of the glibc NSS module **nss-resolve(8)** is required in order to allow glibc's NSS resolver functions to resolve hostnames via **systemd-resolved**.
- Additionally, **systemd-resolved** provides a local DNS stub listener on IP address 127.0.0.53 on the local loopback interface. Programs issuing DNS requests directly, bypassing any local API may be directed to this stub, in order to connect them to **systemd-resolved**. Note however that it is strongly recommended that local programs use the glibc NSS or bus APIs instead (as described above), as various network resolution concepts (such as link-local addressing, or LLMNR Unicode domains) cannot be mapped to the unicast DNS protocol.

The DNS servers contacted are determined from the global settings in /etc/systemd/resolved.conf, the per-link static settings in /etc/systemd/network/*.network files (in case **systemd-networkd.service(8)** is used), the per-link dynamic settings received over DHCP, information provided via **resolvectl(1)**, and any DNS server information made available by other system services. See **resolved.conf(5)** and **systemd.network(5)** for details about systemd's own configuration files for DNS servers. To improve compatibility, /etc/resolv.conf is read in order to discover configured system DNS servers, but only if it is not a symlink to /run/systemd/resolve/stub-resolv.conf, /usr/lib/systemd/resolv.conf or /run/systemd/resolve/resolv.conf (see below).

SYNTHETIC RECORDS

systemd-resolved synthesizes DNS resource records (RRs) for the following cases:

- The local, configured hostname is resolved to all locally configured IP addresses ordered by their scope, or — if none are configured — the IPv4 address 127.0.0.2 (which is on the local loopback interface) and the IPv6 address ::1 (which is the local host).
- The hostnames "localhost" and "localhost.localdomain" as well as any hostname ending in ".localhost" or ".localhost.localdomain" are resolved to the IP addresses 127.0.0.1 and ::1.
- The hostname "_gateway" is resolved to all current default routing gateway addresses, ordered by their metric. This assigns a stable hostname to the current gateway, useful for referencing it independently of the current network configuration state.
- The hostname "_outbound" is resolved to the local IPv4 and IPv6 addresses that are most likely used for communication with other hosts. This is determined by requesting a routing decision to the configured default gateways from the kernel and then using the local IP addresses selected by this decision. This hostname is only available if there is at least one local default gateway configured. This assigns a stable hostname to the local outbound IP addresses, useful for referencing them

independently of the current network configuration state.

- The mappings defined in /etc/hosts are resolved to their configured addresses and back, but they will not affect lookups for non-address types (like MX). Support for /etc/hosts may be disabled with *ReadEtcHosts=no*, see **resolved.conf(5)**.

PROTOCOLS AND ROUTING

The lookup requests that systemd-resolved.service receives are routed to the available DNS servers, LLMNR, and MulticastDNS interfaces according to the following rules:

- Names for which synthetic records are generated (the local hostname, "localhost" and "localdomain", local gateway, as listed in the previous section) and addresses configured in /etc/hosts are never routed to the network and a reply is sent immediately.
- Single-label names are resolved using LLMNR on all local interfaces where LLMNR is enabled. Lookups for IPv4 addresses are only sent via LLMNR on IPv4, and lookups for IPv6 addresses are only sent via LLMNR on IPv6. Note that lookups for single-label synthesized names are not routed to LLMNR, MulticastDNS or unicast DNS.
- Queries for the address records (A and AAAA) of single-label non-synthesized names are resolved via unicast DNS using search domains. For any interface which defines search domains, such look-ups are routed to the servers defined for that interface, suffixed with each of those search domains. When global search domains are defined, such look-ups are routed to the global servers. For each search domain, queries are performed by suffixing the name with each of the search domains in turn. Additionally, lookup of single-label names via unicast DNS may be enabled with the *ResolveUnicastSingleLabel=yes* setting. The details of which servers are queried and how the final reply is chosen are described below. Note that this means that address queries for single-label names are never sent out to remote DNS servers by default, and resolution is only possible if search domains are defined.
- Multi-label names with the domain suffix ".local" are resolved using MulticastDNS on all local interfaces where MulticastDNS is enabled. As with LLMNR, IPv4 address lookups are sent via IPv4 and IPv6 address lookups are sent via IPv6.
- Queries for multi-label names are routed via unicast DNS on local interfaces that have a DNS server configured, plus the globally configured DNS servers if there are any. Which interfaces are used is determined by the routing logic based on search and route-only domains, described below. Note that by default, lookups for domains with the ".local" suffix are not routed to DNS servers, unless the domain is specified explicitly as routing or search domain for the DNS server and interface. This means that on networks where the ".local" domain is defined in a site-specific DNS server, explicit search or routing domains need to be configured to make lookups work within this DNS domain. Note that these days, it's generally recommended to avoid defining ".local" in a DNS server, as [RFC6762^{\[2\]}](#) reserves this domain for exclusive MulticastDNS use.
- Address lookups (reverse lookups) are routed similarly to multi-label names, with the exception that addresses from the link-local address range are never routed to unicast DNS and are only resolved using LLMNR and MulticastDNS (when enabled).

If lookups are routed to multiple interfaces, the first successful response is returned (thus effectively merging the lookup zones on all matching interfaces). If the lookup failed on all interfaces, the last failing response is returned.

Routing of lookups is determined by the per-interface routing domains (search and route-only) and global search domains. See **systemd.network(5)** and **resolvectl(1)** for a description how those settings are set dynamically and the discussion of *Domains=* in **resolved.conf(5)** for a description of globally configured DNS settings.

The following query routing logic applies for unicast DNS lookups initiated by systemd-resolved.service:

- If a name to look up matches (that is: is equal to or has as suffix) any of the configured routing domains (search or route-only) of any link, or the globally configured DNS settings, "best matching" routing domain is determined: the matching one with the most labels. The query is then

sent to all DNS servers of any links or the globally configured DNS servers associated with this "best matching" routing domain. (Note that more than one link might have this same "best matching" routing domain configured, in which case the query is sent to all of them in parallel).

In case of single-label names, when search domains are defined, the same logic applies, except that the name is first suffixed by each of the search domains in turn. Note that this search logic doesn't apply to any names with at least one dot. Also see the discussion about compatibility with the traditional glibc resolver below.

- If a query does not match any configured routing domain (either per-link or global), it is sent to all DNS servers that are configured on links with the *DefaultRoute*= option set, as well as the globally configured DNS server.
- If there is no link configured as *DefaultRoute*= and no global DNS server configured, one of the compiled-in fallback DNS servers is used.
- Otherwise the unicast DNS query fails, as no suitable DNS servers can be determined.

The *DefaultRoute*= option is a boolean setting configurable with **resolvectl** or in .network files. If not set, it is implicitly determined based on the configured DNS domains for a link: if there's a route-only domain other than ".~.", it defaults to false, otherwise to true.

Effectively this means: in order to support single-label non-synthesized names, define appropriate search domains. In order to preferably route all DNS queries not explicitly matched by routing domain configuration to a specific link, configure a ".~." route-only domain on it. This will ensure that other links will not be considered for these queries (unless they too carry such a routing domain). In order to route all such DNS queries to a specific link only if no other link is preferred, set the *DefaultRoute*= option for the link to true and do not configure a ".~." route-only domain on it. Finally, in order to ensure that a specific link never receives any DNS traffic not matching any of its configured routing domains, set the *DefaultRoute*= option for it to false.

See **org.freedesktop.resolve1(5)** for information about the D-Bus APIs `systemd-resolved` provides.

COMPATIBILITY WITH THE TRADITIONAL GLIBC STUB RESOLVER

This section provides a short summary of differences in the stub resolver implemented by **nss-resolve(8)** together with **systemd-resolved** and the traditional stub resolver implemented in `nss-dns`.

- Some names are always resolved internally (see Synthetic Records above). Traditionally they would be resolved by `nss-files` if provided in /etc/hosts. But note that the details of how a query is constructed are under the control of the client library. `nss-dns` will first try to resolve names using search domains and even if those queries are routed to `systemd-resolved`, it will send them out over the network using the usual rules for multi-label name routing^[3].
- Single-label names are not resolved for A and AAAA records using unicast DNS (unless overridden with *ResolveUnicastSingleLabel*=, see **resolved.conf(5)**). This is similar to the **no-tld-query** option being set in **resolv.conf(5)**.
- Search domains are not used for *sufficing* of multi-label names. (Search domains are nevertheless used for lookup *routing*, for names that were originally specified as single-label or multi-label.) Any name with at least one dot is always interpreted as a FQDN. `nss-dns` would resolve names both as relative (using search domains) and absolute FQDN names. Some names would be resolved as relative first, and after that query has failed, as absolute, while other names would be resolved in opposite order. The *ndots* option in /etc/resolv.conf was used to control how many dots the name needs to have to be resolved as relative first. This stub resolver does not implement this at all: multi-label names are only resolved as FQDNs.^[4]
- This resolver has a notion of the special ".local" domain used for MulticastDNS, and will not route queries with that suffix to unicast DNS servers unless explicitly configured, see above. Also, reverse lookups for link-local addresses are not sent to unicast DNS servers.
- This resolver reads and caches /etc/hosts internally. (In other words, `nss-resolve` replaces `nss-files` in addition to `nss-dns`). Entries in /etc/hosts have highest priority.

- This resolver also implements LLMNR and MulticastDNS in addition to the classic unicast DNS protocol, and will resolve single-label names using LLMNR (when enabled) and names ending in ".local" using MulticastDNS (when enabled).
- Environment variables `$LOCALDOMAIN` and `$RES_OPTIONS` described in **resolv.conf(5)** are not supported currently.

/ETC/RESOLV.CONF

Four modes of handling `/etc/resolv.conf` (see **resolv.conf(5)**) are supported:

- **systemd-resolved** maintains the `/run/systemd/resolve/stub-resolv.conf` file for compatibility with traditional Linux programs. This file may be symlinked from `/etc/resolv.conf`. This file lists the 127.0.0.53 DNS stub (see above) as the only DNS server. It also contains a list of search domains that are in use by `systemd-resolved`. The list of search domains is always kept up-to-date. Note that `/run/systemd/resolve/stub-resolv.conf` should not be used directly by applications, but only through a symlink from `/etc/resolv.conf`. This file may be symlinked from `/etc/resolv.conf` in order to connect all local clients that bypass local DNS APIs to **systemd-resolved** with correct search domains settings. This mode of operation is recommended.
- A static file `/usr/lib/systemd/resolv.conf` is provided that lists the 127.0.0.53 DNS stub (see above) as only DNS server. This file may be symlinked from `/etc/resolv.conf` in order to connect all local clients that bypass local DNS APIs to **systemd-resolved**. This file does not contain any search domains.
- **systemd-resolved** maintains the `/run/systemd/resolve/resolv.conf` file for compatibility with traditional Linux programs. This file may be symlinked from `/etc/resolv.conf` and is always kept up-to-date, containing information about all known DNS servers. Note the file format's limitations: it does not know a concept of per-interface DNS servers and hence only contains system-wide DNS server definitions. Note that `/run/systemd/resolve/resolv.conf` should not be used directly by applications, but only through a symlink from `/etc/resolv.conf`. If this mode of operation is used local clients that bypass any local DNS API will also bypass **systemd-resolved** and will talk directly to the known DNS servers.
- Alternatively, `/etc/resolv.conf` may be managed by other packages, in which case **systemd-resolved** will read it for DNS configuration data. In this mode of operation **systemd-resolved** is consumer rather than provider of this configuration file.

Note that the selected mode of operation for this file is detected fully automatically, depending on whether `/etc/resolv.conf` is a symlink to `/run/systemd/resolve/resolv.conf` or lists 127.0.0.53 as DNS server.

SIGNALS

SIGUSR1

Upon reception of the **SIGUSR1** process signal **systemd-resolved** will dump the contents of all DNS resource record caches it maintains, as well as all feature level information it learnt about configured DNS servers into the system logs.

SIGUSR2

Upon reception of the **SIGUSR2** process signal **systemd-resolved** will flush all caches it maintains. Note that it should normally not be necessary to request this explicitly – except for debugging purposes – as **systemd-resolved** flushes the caches automatically anyway any time the host's network configuration changes. Sending this signal to **systemd-resolved** is equivalent to the **resolvectl flush-caches** command, however the latter is recommended since it operates in a synchronous way.

SIGRTMIN+1

Upon reception of the **SIGRTMIN+1** process signal **systemd-resolved** will forget everything it learnt about the configured DNS servers. Specifically any information about server feature support is flushed out, and the server feature probing logic is restarted on the next request, starting with the most fully featured level. Note that it should normally not be necessary to request this explicitly – except for debugging purposes – as **systemd-resolved** automatically forgets learnt information any time the DNS server configuration changes. Sending this signal to **systemd-resolved** is equivalent to the

resolvectl reset-server-features command, however the latter is recommended since it operates in a synchronous way.

SEE ALSO

systemd(1), resolved.conf(5), dnssec-trust-anchors.d(5), nss-resolve(8), resolvectl(1), resolv.conf(5), hosts(5), systemd.network(5), systemd-networkd.service(8)

NOTES

1. RFC3493
<https://tools.ietf.org/html/rfc3493>
2. RFC6762
<https://tools.ietf.org/html/rfc6762>
3. For example, if /etc/resolv.conf has

```
nameserver 127.0.0.53
search foobar.com barbar.com
```

and we look up "localhost", nss-dns will send the following queries to systemd-resolved listening on 127.0.0.53:53: first "localhost.foobar.com", then "localhost.barbar.com", and finally "localhost". If (hopefully) the first two queries fail, systemd-resolved will synthesize an answer for the third query.

When using nss-dns with any search domains, it is thus crucial to always configure nss-files with higher priority and provide mappings for names that should not be resolved using search domains.

4. There are currently more than 1500 top-level domain names defined, and new ones are added regularly, often using "attractive" names that are also likely to be used locally. Not looking up multi-label names in this fashion avoids fragility in both directions: a valid global name could be obscured by a local name, and resolution of a relative local name could suddenly break when a new top-level domain is created, or when a new subdomain of a top-level domain is registered. Resolving any given name as either relative or absolute avoids this ambiguity.

NAME

systemd-sysctl.service, systemd-sysctl – Configure kernel parameters at boot

SYNOPSIS

/lib/systemd/systemd-sysctl [OPTIONS...] [CONFIGFILE...]

systemd-sysctl.service

DESCRIPTION

systemd-sysctl.service is an early boot service that configures **sysctl(8)** kernel parameters by invoking **/lib/systemd/systemd-sysctl**.

When invoked with no arguments, **/lib/systemd/systemd-sysctl** applies all directives from configuration files listed in **sysctl.d(5)**. If one or more filenames are passed on the command line, only the directives in these files are applied.

In addition, **--prefix=** option may be used to limit which sysctl settings are applied.

See **sysctl.d(5)** for information about the configuration of sysctl settings. After sysctl configuration is changed on disk, it must be written to the files in **/proc/sys/** before it takes effect. It is possible to update specific settings, or simply to reload all configuration, see Examples below.

OPTIONS

--prefix=

Only apply rules with the specified prefix.

--cat-config

Copy the contents of config files to standard output. Before each file, the filename is printed as a comment.

--no-pager

Do not pipe output into a pager.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXAMPLES

Example 1. Reset all sysctl settings

```
systemctl restart systemd-sysctl
```

Example 2. View coredump handler configuration

```
# sysctl kernel.core_pattern  
kernel.core_pattern = |/libexec/abrt-hook-ccpp %s %c %p %u %g %t %P %I
```

Example 3. Update coredump handler configuration

```
# /lib/systemd/systemd-sysctl --prefix kernel.core_pattern
```

This searches all the directories listed in **sysctl.d(5)** for configuration files and writes **/proc/sys/kernel/core_pattern**.

Example 4. Update coredump handler configuration according to a specific file

```
# /lib/systemd/systemd-sysctl 50-coredump.conf
```

This applies all the settings found in **50-coredump.conf**. Either **/etc/sysctl.d/50-coredump.conf**, or **/run/sysctl.d/50-coredump.conf**, or **/usr/lib/sysctl.d/50-coredump.conf** will be used, in the order of preference.

See **sysctl(8)** for various ways to directly apply sysctl settings.

SEE ALSO

systemd(1), sysctl.d(5), sysctl(8),

NAME

`systemd-system.conf`, `system.conf.d`, `systemd-user.conf`, `user.conf.d` – System and session service manager configuration files

SYNOPSIS

```
/etc/systemd/system.conf, /etc/systemd/system.conf.d/*.conf, /run/systemd/system.conf.d/*.conf,
/lib/systemd/system.conf.d/*.conf
~/.config/systemd/user.conf, /etc/systemd/user.conf, /etc/systemd/user.conf.d/*.conf,
/run/systemd/user.conf.d/*.conf, /usr/lib/systemd/user.conf.d/*.conf
```

DESCRIPTION

When run as a system instance, **systemd** interprets the configuration file `system.conf` and the files in `system.conf.d` directories; when run as a user instance, it interprets the configuration file `user.conf` (either in the home directory of the user, or if not found, under `/etc/systemd/`) and the files in `user.conf.d` directories. These configuration files contain a few settings controlling basic manager operations.

See **systemd.syntax(7)** for a general description of the syntax.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in `/etc/systemd/` contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from `/usr/lib/systemd/*.conf.d/`, `/usr/local/lib/systemd/*.conf.d/`, and `/etc/systemd/*.conf.d/`. Those drop-ins have higher precedence and override the main configuration file. Files in the `*.conf.d/` configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under `/usr/`. Files in `/etc/` are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to `/dev/null` in the configuration directory in `/etc/`, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the [Manager] section:

LogColor=, *LogLevel=*, *LogLocation=*, *LogTarget=*, *LogTime=*, *DumpCore=yes*, *CrashChangeVT=no*,
CrashShell=no, *CrashReboot=no*, *ShowStatus=yes*, *DefaultStandardOutput=journal*,
DefaultStandardError=inherit

Configures various parameters of basic manager operation. These options may be overridden by the respective process and kernel command line arguments. See **systemd(1)** for details.

CtrlAltDelBurstAction=

Defines what action will be performed if user presses Ctrl–Alt–Delete more than 7 times in 2s. Can be set to "reboot–force", "poweroff–force", "reboot–immediate", "poweroff–immediate" or disabled with "none". Defaults to "reboot–force".

CPUAffinity=

Configures the CPU affinity for the service manager as well as the default CPU affinity for all forked off processes. Takes a list of CPU indices or ranges separated by either whitespace or commas. CPU ranges are specified by the lower and upper CPU indices separated by a dash. This option may be specified more than once, in which case the specified CPU affinity masks are merged. If the empty

string is assigned, the mask is reset, all assignments prior to this will have no effect. Individual services may override the CPU affinity for their processes with the *CPUAffinity*= setting in unit files, see **systemd.exec(5)**.

***NUMAPolicy*=**

Configures the NUMA memory policy for the service manager and the default NUMA memory policy for all forked off processes. Individual services may override the default policy with the *NUMAPolicy*= setting in unit files, see **systemd.exec(5)**.

***NUMAMask*=**

Configures the NUMA node mask that will be associated with the selected NUMA policy. Note that **default** and **local** NUMA policies don't require explicit NUMA node mask and value of the option can be empty. Similarly to *NUMAPolicy*=, value can be overridden by individual services in unit files, see **systemd.exec(5)**.

***RuntimeWatchdogSec*=, *RebootWatchdogSec*=, *KExecWatchdogSec*=**

Configure the hardware watchdog at runtime and at reboot. Takes a timeout value in seconds (or in other time units if suffixed with "ms", "min", "h", "d", "w"). If *RuntimeWatchdogSec*= is set to a non-zero value, the watchdog hardware (/dev/watchdog or the path specified with *WatchdogDevice*= or the kernel option *systemd.watchdog-device*) will be programmed to automatically reboot the system if it is not contacted within the specified timeout interval. The system manager will ensure to contact it at least once in half the specified timeout interval. This feature requires a hardware watchdog device to be present, as it is commonly the case in embedded and server systems. Not all hardware watchdogs allow configuration of all possible reboot timeout values, in which case the closest available timeout is picked. *RebootWatchdogSec*= may be used to configure the hardware watchdog when the system is asked to reboot. It works as a safety net to ensure that the reboot takes place even if a clean reboot attempt times out. Note that the *RebootWatchdogSec*= timeout applies only to the second phase of the reboot, i.e. after all regular services are already terminated, and after the system and service manager process (PID 1) got replaced by the **systemd-shutdown** binary, see **system bootup(7)** for details. During the first phase of the shutdown operation the system and service manager remains running and hence *RuntimeWatchdogSec*= is still honoured. In order to define a timeout on this first phase of system shutdown, configure *JobTimeoutSec*= and *JobTimeoutAction*= in the [Unit] section of the shutdown.target unit. By default *RuntimeWatchdogSec*= defaults to 0 (off), and *RebootWatchdogSec*= to 10min. *KExecWatchdogSec*= may be used to additionally enable the watchdog when kexec is being executed rather than when rebooting. Note that if the kernel does not reset the watchdog on kexec (depending on the specific hardware and/or driver), in this case the watchdog might not get disabled after kexec succeeds and thus the system might get rebooted, unless *RuntimeWatchdogSec*= is also enabled at the same time. For this reason it is recommended to enable *KExecWatchdogSec*= only if *RuntimeWatchdogSec*= is also enabled. These settings have no effect if a hardware watchdog is not available.

***WatchdogDevice*=**

Configure the hardware watchdog device that the runtime and shutdown watchdog timers will open and use. Defaults to /dev/watchdog. This setting has no effect if a hardware watchdog is not available.

***CapabilityBoundingSet*=**

Controls which capabilities to include in the capability bounding set for PID 1 and its children. See **capabilities(7)** for details. Takes a whitespace-separated list of capability names as read by **cap_from_name(3)**. Capabilities listed will be included in the bounding set, all others are removed. If the list of capabilities is prefixed with ~, all but the listed capabilities will be included, the effect of the assignment inverted. Note that this option also affects the respective capabilities in the effective, permitted and inheritable capability sets. The capability bounding set may also be individually configured for units using the *CapabilityBoundingSet*= directive for units, but note that capabilities dropped for PID 1 cannot be regained in individual units, they are lost for good.

***NoNewPrivileges*=**

Takes a boolean argument. If true, ensures that PID 1 and all its children can never gain new privileges through **execve(2)** (e.g. via setuid or setgid bits, or filesystem capabilities). Defaults to false. General

purpose distributions commonly rely on executables with setuid or setgid bits and will thus not function properly with this option enabled. Individual units cannot disable this option. Also see **No New Privileges Flag**^[1].

SystemCallArchitectures=

Takes a space-separated list of architecture identifiers. Selects from which architectures system calls may be invoked on this system. This may be used as an effective way to disable invocation of non-native binaries system-wide, for example to prohibit execution of 32-bit x86 binaries on 64-bit x86-64 systems. This option operates system-wide, and acts similar to the *SystemCallArchitectures*= setting of unit files, see **systemd.exec(5)** for details. This setting defaults to the empty list, in which case no filtering of system calls based on architecture is applied. Known architecture identifiers are "x86", "x86-64", "x32", "arm" and the special identifier "native". The latter implicitly maps to the native architecture of the system (or more specifically, the architecture the system manager was compiled for). Set this setting to "native" to prohibit execution of any non-native binaries. When a binary executes a system call of an architecture that is not listed in this setting, it will be immediately terminated with the SIGSYS signal.

TimerSlackNSec=

Sets the timer slack in nanoseconds for PID 1, which is inherited by all executed processes, unless overridden individually, for example with the *TimerSlackNSec*= setting in service units (for details see **systemd.exec(5)**). The timer slack controls the accuracy of wake-ups triggered by system timers. See **prctl(2)** for more information. Note that in contrast to most other time span definitions this parameter takes an integer value in nano-seconds if no unit is specified. The usual time units are understood too.

StatusUnitFormat=

Takes **name**, **description** or **combined** as the value. If **name**, the system manager will use unit names in status messages (e.g. "systemd-journald.service"), instead of the longer and more informative descriptions set with *Description*= (e.g. "Journal Logging Service"). If **combined**, the system manager will use both unit names and descriptions in status messages (e.g. "systemd-journald.service – Journal Logging Service").

See **systemd.unit(5)** for details about unit names and *Description*=.

DefaultTimerAccuracySec=

Sets the default accuracy of timer units. This controls the global default for the *AccuracySec*= setting of timer units, see **systemd.timer(5)** for details. *AccuracySec*= set in individual units override the global default for the specific unit. Defaults to 1min. Note that the accuracy of timer units is also affected by the configured timer slack for PID 1, see *TimerSlackNSec*= above.

DefaultTimeoutStartSec=, *DefaultTimeoutStopSec*=, *DefaultTimeoutAbortSec*=, *DefaultRestartSec*=

Configures the default timeouts for starting, stopping and aborting of units, as well as the default time to sleep between automatic restarts of units, as configured per-unit in *TimeoutStartSec*=, *TimeoutStopSec*=, *TimeoutAbortSec*= and *RestartSec*= (for services, see **systemd.service(5)** for details on the per-unit settings). Disabled by default, when service with *Type=oneshot* is used. For non-service units, *DefaultTimeoutStartSec*= sets the default *TimeoutSec*= value. *DefaultTimeoutStartSec*= and *DefaultTimeoutStopSec*= default to 90s. *DefaultTimeoutAbortSec*= is not set by default so that all units fall back to *TimeoutStopSec*=. *DefaultRestartSec*= defaults to 100ms.

DefaultStartLimitIntervalSec=, *DefaultStartLimitBurst*=

Configure the default unit start rate limiting, as configured per-service by *StartLimitIntervalSec*= and *StartLimitBurst*=. See **systemd.service(5)** for details on the per-service settings.

DefaultStartLimitIntervalSec= defaults to 10s. *DefaultStartLimitBurst*= defaults to 5.

DefaultEnvironment=

Configures environment variables passed to all executed processes. Takes a space-separated list of variable assignments. See **environ(7)** for details about environment variables.

Simple "%"–specifier expansion is supported, see below for a list of supported specifiers.

Example:

```
DefaultEnvironment="VAR1=word1 word2" VAR2=word3 "VAR3=word 5 6"
```

Sets three variables "VAR1", "VAR2", "VAR3".

ManagerEnvironment=

Takes the same arguments as *DefaultEnvironment*=, see above. Sets environment variables just for the manager process itself. In contrast to user managers, these variables are not inherited by processes spawned by the system manager, use *DefaultEnvironment*= for that. Note that these variables are merged into the existing environment block. In particular, in case of the system manager, this includes variables set by the kernel based on the kernel command line.

Setting environment variables for the manager process may be useful to modify its behaviour. See [ENVIRONMENT](#)^[2] for a descriptions of some variables understood by **systemd**.

Simple "%"–specifier expansion is supported, see below for a list of supported specifiers.

DefaultCPUAccounting=, *DefaultBlockIOAccounting*=, *DefaultMemoryAccounting*=,
DefaultTasksAccounting=, *DefaultIOAccounting*=, *DefaultIPAccounting*=

Configure the default resource accounting settings, as configured per–unit by *CPUAccounting*=, *BlockIOAccounting*=, *MemoryAccounting*=, *TasksAccounting*=, *IOAccounting*= and *IPAccounting*=. See [systemd.resource-control\(5\)](#) for details on the per–unit settings. *DefaultTasksAccounting*= defaults to yes, *DefaultMemoryAccounting*= to yes. *DefaultCPUAccounting*= defaults to yes if enabling CPU accounting doesn't require the CPU controller to be enabled (Linux 4.15+ using the unified hierarchy for resource control), otherwise it defaults to no. The other three settings default to no.

DefaultTasksMax=

Configure the default value for the per–unit *TasksMax*= setting. See [systemd.resource-control\(5\)](#) for details. This setting applies to all unit types that support resource control settings, with the exception of slice units. Defaults to 15% of the minimum of *kernel.pid_max*=, *kernel.threads_max*= and root cgroup *pids.max*. Kernel has a default value for *kernel.pid_max*= and an algorithm of counting in case of more than 32 cores. For example with the default *kernel.pid_max*=, *DefaultTasksMax*= defaults to 4915, but might be greater in other systems or smaller in OS containers.

DefaultLimitCPU=, *DefaultLimitFSIZE*=, *DefaultLimitDATA*=, *DefaultLimitSTACK*=,
DefaultLimitCORE=, *DefaultLimitRSS*=, *DefaultLimitNOFILE*=, *DefaultLimitAS*=,
DefaultLimitNPROC=, *DefaultLimitMEMLOCK*=, *DefaultLimitLOCKS*=, *DefaultLimitSIGPENDING*=,
DefaultLimitMSGQUEUE=, *DefaultLimitNICE*=, *DefaultLimitRTPRIO*=, *DefaultLimitRTTIME*=

These settings control various default resource limits for processes executed by units. See [setrlimit\(2\)](#) for details. These settings may be overridden in individual units using the corresponding *LimitXXX*= directives and they accept the same parameter syntax, see [systemd.exec\(5\)](#) for details. Note that these resource limits are only defaults for units, they are not applied to the service manager process (i.e. PID 1) itself.

Most of these settings are unset, which means the resource limits are inherited from the kernel or, if invoked in a container, from the container manager. However, the following have defaults:

- *DefaultLimitNOFILE*= defaults to "1024:524288".
- *DefaultLimitCORE*= does not have a default but it is worth mentioning that *RLIMIT_CORE* is set to "infinity" by PID 1 which is inherited by its children.
- Note that the service manager internally increases *RLIMIT_MEMLOCK* for itself, however the limit is reverted to the original value for child processes forked off.

DefaultOOMPolicy=

Configure the default policy for reacting to processes being killed by the Linux Out–Of–Memory

(OOM) killer. This may be used to pick a global default for the per-unit *OOMPpolicy=* setting. See **systemd.service(5)** for details. Note that this default is not used for services that have *Delegate=* turned on.

SPECIFIERS

Specifiers may be used in the *DefaultEnvironment=* and *ManagerEnvironment=* settings. The following expansions are understood:

Table 1. Specifiers available

SEE ALSO

systemd(1), systemd.directives(7), systemd.exec(5), systemd.service(5), environ(7), capabilities(7)

NOTES

1. No New Privileges Flag
https://www.kernel.org/doc/html/latest/userspace-api/no_new_privs.html

2. ENVIRONMENT
<https://systemd.io/ENVIRONMENT>

NAME

systemd-mount, **systemd-umount** – Establish and destroy transient mount or auto-mount points

SYNOPSIS

```
systemd-mount [OPTIONS...] WHAT [WHERE]
systemd-mount [OPTIONS...] --list
systemd-mount [OPTIONS...] --umount WHAT/WHERE...
```

DESCRIPTION

systemd-mount may be used to create and start a transient .mount or .automount unit of the file system *WHAT* on the mount point *WHERE*.

In many ways, **systemd-mount** is similar to the lower-level **mount**(8) command, however instead of executing the mount operation directly and immediately, **systemd-mount** schedules it through the service manager job queue, so that it may pull in further dependencies (such as parent mounts, or a file system checker to execute a priori), and may make use of the auto-mounting logic.

The command takes either one or two arguments. If only one argument is specified it should refer to a block device or regular file containing a file system (e.g. "/dev/sdb1" or "/path/to/disk.img"). The block device or image file is then probed for a file system label and other metadata, and is mounted to a directory below /run/media/system/ whose name is generated from the file system label. In this mode the block device or image file must exist at the time of invocation of the command, so that it may be probed. If the device is found to be a removable block device (e.g. a USB stick), an automount point is created instead of a regular mount point (i.e. the **--automount**= option is implied, see below).

If two arguments are specified, the first indicates the mount source (the *WHAT*) and the second indicates the path to mount it on (the *WHERE*). In this mode no probing of the source is attempted, and a backing device node doesn't have to exist. However, if this mode is combined with **--discover**, device node probing for additional metadata is enabled, and – much like in the single-argument case discussed above – the specified device has to exist at the time of invocation of the command.

Use the **--list** command to show a terse table of all local, known block devices with file systems that may be mounted with this command.

systemd-umount can be used to unmount a mount or automount point. It is the same as **systemd-mount** **--umount**.

OPTIONS

The following options are understood:

--no-block

Do not synchronously wait for the requested operation to finish. If this is not specified, the job will be verified, enqueued and **systemd-mount** will wait until the mount or automount unit's start-up is completed. By passing this argument, it is only verified and enqueued.

-l, --full

Do not ellipsize the output when **--list** is specified.

--no-pager

Do not pipe output into a pager.

--no-legend

Do not print the legend, i.e. column headers and the footer with hints.

--no-ask-password

Do not query the user for authentication for privileged operations.

--quiet, -q

Suppresses additional informational output while running.

--discover

Enable probing of the mount source. This switch is implied if a single argument is specified on the command line. If passed, additional metadata is read from the device to enhance the unit to create. For

example, a descriptive string for the transient units is generated from the file system label and device model. Moreover if a removable block device (e.g. USB stick) is detected an automount unit instead of a regular mount unit is created, with a short idle timeout, in order to ensure the file-system is placed in a clean state quickly after each access.

--type=, -t

Specifies the file system type to mount (e.g. "vfat" or "ext4"). If omitted or set to "auto", the file system type is determined automatically.

--options=, -o

Additional mount options for the mount point.

--owner=USER

Let the specified user *USER* own the mounted file system. This is done by appending **uid=** and **gid=** options to the list of mount options. Only certain file systems support this option.

--fsck=

Takes a boolean argument, defaults to on. Controls whether to run a file system check immediately before the mount operation. In the automount case (see **--automount=** below) the check will be run the moment the first access to the device is made, which might slightly delay the access.

--description=

Provide a description for the mount or automount unit. See *Description=* in **systemd.unit(5)**.

--property=, -p

Sets a unit property for the mount unit that is created. This takes an assignment in the same format as **systemctl(1)**'s **set-property** command.

--automount=

Takes a boolean argument. Controls whether to create an automount point or a regular mount point. If true an automount point is created that is backed by the actual file system at the time of first access. If false a plain mount point is created that is backed by the actual file system immediately. Automount points have the benefit that the file system stays unmounted and hence in clean state until it is first accessed. In automount mode the **--timeout-idle-sec=** switch (see below) may be used to ensure the mount point is unmounted automatically after the last access and an idle period passed.

If this switch is not specified it defaults to false. If not specified and **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, it is set to true, in order to increase the chance that the file system is in a fully clean state if the device is unplugged abruptly.

-A

Equivalent to **--automount=yes**.

--timeout-idle-sec=

Takes a time value that controls the idle timeout in automount mode. If set to "infinity" (the default) no automatic unmounts are done. Otherwise the file system backing the automount point is detached after the last access and the idle timeout passed. See **systemd.time(7)** for details on the time syntax supported. This option has no effect if only a regular mount is established, and automounting is not used.

Note that if **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, **--timeout-idle-sec=1s** is implied.

--automount-property=

Similar to **--property=**, but applies additional properties to the automount unit created, instead of the mount unit.

--bind-device

This option only has an effect in automount mode, and controls whether the automount unit shall be

bound to the backing device's lifetime. If set, the automount point will be removed automatically when the backing device vanishes. By default the automount point stays around, and subsequent accesses will block until backing device is replugged. This option has no effect in case of non-device mounts, such as network or virtual file system mounts.

Note that if **--discover** is used (or only a single argument passed, which implies **--discover**, see above), and the file system block device is detected to be removable, this option is implied.

--list

Instead of establishing a mount or automount point, print a terse list of block devices containing file systems that may be mounted with "systemd-mount", along with useful metadata such as labels, etc.

-u, --umount

Stop the mount and automount units corresponding to the specified mount points *WHERE* or the devices *WHAT*. **systemd-mount** with this option or **systemd-umount** can take multiple arguments which can be mount points, devices, /etc/fstab style node names, or backing files corresponding to loop devices, like **systemd-mount --umount /path/to/umount /dev/sda1 UUID=xxxxxx-xxxx LABEL=xxxxx /path/to/disk.img**. Note that when **-H** or **-M** is specified, only absolute paths to mount points are supported.

-G, --collect

Unload the transient unit after it completed, even if it failed. Normally, without this option, all mount units that mount and failed are kept in memory until the user explicitly resets their failure state with **systemctl reset-failed** or an equivalent command. On the other hand, units that stopped successfully are unloaded immediately. If this option is turned on the "garbage collection" of units is more aggressive, and unloads units regardless if they exited successfully or failed. This option is a shortcut for **--property=CollectMode=inactive-or-failed**, see the explanation for *CollectMode*= in **systemd.unit(5)** for further information.

--user

Talk to the service manager of the calling user, rather than the service manager of the system.

--system

Talk to the service manager of the system. This is the implied default.

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":"; and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

THE UDEV DATABASE

If **--discover** is used, **systemd-mount** honors a couple of additional udev properties of block devices:

SYSTEMD_MOUNT_OPTIONS=

The mount options to use, if **--options=** is not used.

SYSTEMD_MOUNT_WHERE=

The file system path to place the mount point at, instead of the automatically generated one.

EXAMPLE

Use a udev rule like the following to automatically mount all USB storage plugged in:

```
ACTION=="add", SUBSYSTEMS=="usb", SUBSYSTEM=="block", ENV{ID_FS_USAGE}=="filesystem", \
    RUN{program}+="/usr/bin/systemd-mount --no-block --automount=yes --collect $devnode"
```

SEE ALSO

systemd(1), mount(8), systemctl(1), systemd.unit(5), systemd.mount(5), systemd.automount(5),
systemd-run(1)

NAME

`systemd-system.conf`, `system.conf.d`, `systemd-user.conf`, `user.conf.d` – System and session service manager configuration files

SYNOPSIS

```
/etc/systemd/system.conf, /etc/systemd/system.conf.d/*.conf, /run/systemd/system.conf.d/*.conf,
/lib/systemd/system.conf.d/*.conf
~/.config/systemd/user.conf, /etc/systemd/user.conf, /etc/systemd/user.conf.d/*.conf,
/run/systemd/user.conf.d/*.conf, /usr/lib/systemd/user.conf.d/*.conf
```

DESCRIPTION

When run as a system instance, **systemd** interprets the configuration file `system.conf` and the files in `system.conf.d` directories; when run as a user instance, it interprets the configuration file `user.conf` (either in the home directory of the user, or if not found, under `/etc/systemd/`) and the files in `user.conf.d` directories. These configuration files contain a few settings controlling basic manager operations.

See **systemd.syntax(7)** for a general description of the syntax.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in `/etc/systemd/` contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from `/usr/lib/systemd/*.conf.d/`, `/usr/local/lib/systemd/*.conf.d/`, and `/etc/systemd/*.conf.d/`. Those drop-ins have higher precedence and override the main configuration file. Files in the `*.conf.d/` configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under `/usr/`. Files in `/etc/` are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to `/dev/null` in the configuration directory in `/etc/`, with the same filename as the vendor configuration file.

OPTIONS

All options are configured in the [Manager] section:

LogColor=, *LogLevel=*, *LogLocation=*, *LogTarget=*, *LogTime=*, *DumpCore=yes*, *CrashChangeVT=no*,
CrashShell=no, *CrashReboot=no*, *ShowStatus=yes*, *DefaultStandardOutput=journal*,
DefaultStandardError=inherit

Configures various parameters of basic manager operation. These options may be overridden by the respective process and kernel command line arguments. See **systemd(1)** for details.

CtrlAltDelBurstAction=

Defines what action will be performed if user presses Ctrl–Alt–Delete more than 7 times in 2s. Can be set to "reboot–force", "poweroff–force", "reboot–immediate", "poweroff–immediate" or disabled with "none". Defaults to "reboot–force".

CPUAffinity=

Configures the CPU affinity for the service manager as well as the default CPU affinity for all forked off processes. Takes a list of CPU indices or ranges separated by either whitespace or commas. CPU ranges are specified by the lower and upper CPU indices separated by a dash. This option may be specified more than once, in which case the specified CPU affinity masks are merged. If the empty

string is assigned, the mask is reset, all assignments prior to this will have no effect. Individual services may override the CPU affinity for their processes with the *CPUAffinity*= setting in unit files, see **systemd.exec(5)**.

***NUMAPolicy*=**

Configures the NUMA memory policy for the service manager and the default NUMA memory policy for all forked off processes. Individual services may override the default policy with the *NUMAPolicy*= setting in unit files, see **systemd.exec(5)**.

***NUMAMask*=**

Configures the NUMA node mask that will be associated with the selected NUMA policy. Note that **default** and **local** NUMA policies don't require explicit NUMA node mask and value of the option can be empty. Similarly to *NUMAPolicy*=, value can be overridden by individual services in unit files, see **systemd.exec(5)**.

***RuntimeWatchdogSec*=, *RebootWatchdogSec*=, *KExecWatchdogSec*=**

Configure the hardware watchdog at runtime and at reboot. Takes a timeout value in seconds (or in other time units if suffixed with "ms", "min", "h", "d", "w"). If *RuntimeWatchdogSec*= is set to a non-zero value, the watchdog hardware (/dev/watchdog or the path specified with *WatchdogDevice*= or the kernel option *systemd.watchdog-device*) will be programmed to automatically reboot the system if it is not contacted within the specified timeout interval. The system manager will ensure to contact it at least once in half the specified timeout interval. This feature requires a hardware watchdog device to be present, as it is commonly the case in embedded and server systems. Not all hardware watchdogs allow configuration of all possible reboot timeout values, in which case the closest available timeout is picked. *RebootWatchdogSec*= may be used to configure the hardware watchdog when the system is asked to reboot. It works as a safety net to ensure that the reboot takes place even if a clean reboot attempt times out. Note that the *RebootWatchdogSec*= timeout applies only to the second phase of the reboot, i.e. after all regular services are already terminated, and after the system and service manager process (PID 1) got replaced by the **systemd-shutdown** binary, see **system bootup(7)** for details. During the first phase of the shutdown operation the system and service manager remains running and hence *RuntimeWatchdogSec*= is still honoured. In order to define a timeout on this first phase of system shutdown, configure *JobTimeoutSec*= and *JobTimeoutAction*= in the [Unit] section of the shutdown.target unit. By default *RuntimeWatchdogSec*= defaults to 0 (off), and *RebootWatchdogSec*= to 10min. *KExecWatchdogSec*= may be used to additionally enable the watchdog when kexec is being executed rather than when rebooting. Note that if the kernel does not reset the watchdog on kexec (depending on the specific hardware and/or driver), in this case the watchdog might not get disabled after kexec succeeds and thus the system might get rebooted, unless *RuntimeWatchdogSec*= is also enabled at the same time. For this reason it is recommended to enable *KExecWatchdogSec*= only if *RuntimeWatchdogSec*= is also enabled. These settings have no effect if a hardware watchdog is not available.

***WatchdogDevice*=**

Configure the hardware watchdog device that the runtime and shutdown watchdog timers will open and use. Defaults to /dev/watchdog. This setting has no effect if a hardware watchdog is not available.

***CapabilityBoundingSet*=**

Controls which capabilities to include in the capability bounding set for PID 1 and its children. See **capabilities(7)** for details. Takes a whitespace-separated list of capability names as read by **cap_from_name(3)**. Capabilities listed will be included in the bounding set, all others are removed. If the list of capabilities is prefixed with ~, all but the listed capabilities will be included, the effect of the assignment inverted. Note that this option also affects the respective capabilities in the effective, permitted and inheritable capability sets. The capability bounding set may also be individually configured for units using the *CapabilityBoundingSet*= directive for units, but note that capabilities dropped for PID 1 cannot be regained in individual units, they are lost for good.

***NoNewPrivileges*=**

Takes a boolean argument. If true, ensures that PID 1 and all its children can never gain new privileges through **execve(2)** (e.g. via setuid or setgid bits, or filesystem capabilities). Defaults to false. General

purpose distributions commonly rely on executables with setuid or setgid bits and will thus not function properly with this option enabled. Individual units cannot disable this option. Also see **No New Privileges Flag**^[1].

SystemCallArchitectures=

Takes a space-separated list of architecture identifiers. Selects from which architectures system calls may be invoked on this system. This may be used as an effective way to disable invocation of non-native binaries system-wide, for example to prohibit execution of 32-bit x86 binaries on 64-bit x86-64 systems. This option operates system-wide, and acts similar to the *SystemCallArchitectures*= setting of unit files, see **systemd.exec(5)** for details. This setting defaults to the empty list, in which case no filtering of system calls based on architecture is applied. Known architecture identifiers are "x86", "x86-64", "x32", "arm" and the special identifier "native". The latter implicitly maps to the native architecture of the system (or more specifically, the architecture the system manager was compiled for). Set this setting to "native" to prohibit execution of any non-native binaries. When a binary executes a system call of an architecture that is not listed in this setting, it will be immediately terminated with the SIGSYS signal.

TimerSlackNSec=

Sets the timer slack in nanoseconds for PID 1, which is inherited by all executed processes, unless overridden individually, for example with the *TimerSlackNSec*= setting in service units (for details see **systemd.exec(5)**). The timer slack controls the accuracy of wake-ups triggered by system timers. See **prctl(2)** for more information. Note that in contrast to most other time span definitions this parameter takes an integer value in nano-seconds if no unit is specified. The usual time units are understood too.

StatusUnitFormat=

Takes **name**, **description** or **combined** as the value. If **name**, the system manager will use unit names in status messages (e.g. "systemd-journald.service"), instead of the longer and more informative descriptions set with *Description*= (e.g. "Journal Logging Service"). If **combined**, the system manager will use both unit names and descriptions in status messages (e.g. "systemd-journald.service – Journal Logging Service").

See **systemd.unit(5)** for details about unit names and *Description*=.

DefaultTimerAccuracySec=

Sets the default accuracy of timer units. This controls the global default for the *AccuracySec*= setting of timer units, see **systemd.timer(5)** for details. *AccuracySec*= set in individual units override the global default for the specific unit. Defaults to 1min. Note that the accuracy of timer units is also affected by the configured timer slack for PID 1, see *TimerSlackNSec*= above.

DefaultTimeoutStartSec=, *DefaultTimeoutStopSec*=, *DefaultTimeoutAbortSec*=, *DefaultRestartSec*=

Configures the default timeouts for starting, stopping and aborting of units, as well as the default time to sleep between automatic restarts of units, as configured per-unit in *TimeoutStartSec*=, *TimeoutStopSec*=, *TimeoutAbortSec*= and *RestartSec*= (for services, see **systemd.service(5)** for details on the per-unit settings). Disabled by default, when service with *Type=oneshot* is used. For non-service units, *DefaultTimeoutStartSec*= sets the default *TimeoutSec*= value. *DefaultTimeoutStartSec*= and *DefaultTimeoutStopSec*= default to 90s. *DefaultTimeoutAbortSec*= is not set by default so that all units fall back to *TimeoutStopSec*=. *DefaultRestartSec*= defaults to 100ms.

DefaultStartLimitIntervalSec=, *DefaultStartLimitBurst*=

Configure the default unit start rate limiting, as configured per-service by *StartLimitIntervalSec*= and *StartLimitBurst*=. See **systemd.service(5)** for details on the per-service settings.

DefaultStartLimitIntervalSec= defaults to 10s. *DefaultStartLimitBurst*= defaults to 5.

DefaultEnvironment=

Configures environment variables passed to all executed processes. Takes a space-separated list of variable assignments. See **environ(7)** for details about environment variables.

Simple "%"–specifier expansion is supported, see below for a list of supported specifiers.

Example:

```
DefaultEnvironment="VAR1=word1 word2" VAR2=word3 "VAR3=word 5 6"
```

Sets three variables "VAR1", "VAR2", "VAR3".

ManagerEnvironment=

Takes the same arguments as *DefaultEnvironment*=, see above. Sets environment variables just for the manager process itself. In contrast to user managers, these variables are not inherited by processes spawned by the system manager, use *DefaultEnvironment*= for that. Note that these variables are merged into the existing environment block. In particular, in case of the system manager, this includes variables set by the kernel based on the kernel command line.

Setting environment variables for the manager process may be useful to modify its behaviour. See [ENVIRONMENT](#)^[2] for a descriptions of some variables understood by **systemd**.

Simple "%"–specifier expansion is supported, see below for a list of supported specifiers.

DefaultCPUAccounting=, *DefaultBlockIOAccounting*=, *DefaultMemoryAccounting*=,
DefaultTasksAccounting=, *DefaultIOAccounting*=, *DefaultIPAccounting*=

Configure the default resource accounting settings, as configured per–unit by *CPUAccounting*=, *BlockIOAccounting*=, *MemoryAccounting*=, *TasksAccounting*=, *IOAccounting*= and *IPAccounting*=. See [systemd.resource-control\(5\)](#) for details on the per–unit settings. *DefaultTasksAccounting*= defaults to yes, *DefaultMemoryAccounting*= to yes. *DefaultCPUAccounting*= defaults to yes if enabling CPU accounting doesn't require the CPU controller to be enabled (Linux 4.15+ using the unified hierarchy for resource control), otherwise it defaults to no. The other three settings default to no.

DefaultTasksMax=

Configure the default value for the per–unit *TasksMax*= setting. See [systemd.resource-control\(5\)](#) for details. This setting applies to all unit types that support resource control settings, with the exception of slice units. Defaults to 15% of the minimum of *kernel.pid_max*=, *kernel.threads_max*= and root cgroup *pids.max*. Kernel has a default value for *kernel.pid_max*= and an algorithm of counting in case of more than 32 cores. For example with the default *kernel.pid_max*=, *DefaultTasksMax*= defaults to 4915, but might be greater in other systems or smaller in OS containers.

DefaultLimitCPU=, *DefaultLimitFSIZE*=, *DefaultLimitDATA*=, *DefaultLimitSTACK*=,
DefaultLimitCORE=, *DefaultLimitRSS*=, *DefaultLimitNOFILE*=, *DefaultLimitAS*=,
DefaultLimitNPROC=, *DefaultLimitMEMLOCK*=, *DefaultLimitLOCKS*=, *DefaultLimitSIGPENDING*=,
DefaultLimitMSGQUEUE=, *DefaultLimitNICE*=, *DefaultLimitRTPRIO*=, *DefaultLimitRTTIME*=

These settings control various default resource limits for processes executed by units. See [setrlimit\(2\)](#) for details. These settings may be overridden in individual units using the corresponding *LimitXXX*= directives and they accept the same parameter syntax, see [systemd.exec\(5\)](#) for details. Note that these resource limits are only defaults for units, they are not applied to the service manager process (i.e. PID 1) itself.

Most of these settings are unset, which means the resource limits are inherited from the kernel or, if invoked in a container, from the container manager. However, the following have defaults:

- *DefaultLimitNOFILE*= defaults to "1024:524288".
- *DefaultLimitCORE*= does not have a default but it is worth mentioning that *RLIMIT_CORE* is set to "infinity" by PID 1 which is inherited by its children.
- Note that the service manager internally increases *RLIMIT_MEMLOCK* for itself, however the limit is reverted to the original value for child processes forked off.

DefaultOOMPolicy=

Configure the default policy for reacting to processes being killed by the Linux Out–Of–Memory

(OOM) killer. This may be used to pick a global default for the per-unit *OOMPpolicy=* setting. See **systemd.service(5)** for details. Note that this default is not used for services that have *Delegate=* turned on.

SPECIFIERS

Specifiers may be used in the *DefaultEnvironment=* and *ManagerEnvironment=* settings. The following expansions are understood:

Table 1. Specifiers available

SEE ALSO

systemd(1), systemd.directives(7), systemd.exec(5), systemd.service(5), environ(7), capabilities(7)

NOTES

1. No New Privileges Flag
https://www.kernel.org/doc/html/latest/userspace-api/no_new_privs.html

2. ENVIRONMENT
<https://systemd.io/ENVIRONMENT>

NAME

systemd, init – systemd system and service manager

SYNOPSIS

/lib/systemd/systemd [OPTIONS...]

init [OPTIONS...] {COMMAND}

DESCRIPTION

systemd is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. Separate instances are started for logged-in users to start their services.

systemd is usually not invoked directly by the user, but is installed as the `/sbin/init` symlink and started during early boot. The user manager instances are started automatically through the **user@.service(5)** service.

For compatibility with SysV, if the binary is called as **init** and is not the first process on the machine (PID is not 1), it will execute **telinit** and pass all command line arguments unmodified. That means **init** and **telinit** are mostly equivalent when invoked from normal login sessions. See **telinit(8)** for more information.

When run as a system instance, systemd interprets the configuration file `system.conf` and the files in `system.conf.d` directories; when run as a user instance, systemd interprets the configuration file `user.conf` and the files in `user.conf.d` directories. See **systemd-system.conf(5)** for more information.

CONCEPTS

systemd provides a dependency system between various entities called "units" of 11 different types. Units encapsulate various objects that are relevant for system boot-up and maintenance. The majority of units are configured in unit configuration files, whose syntax and basic set of options is described in **systemd.unit(5)**, however some are created automatically from other configuration files, dynamically from system state or programmatically at runtime. Units may be "active" (meaning started, bound, plugged in, ..., depending on the unit type, see below), or "inactive" (meaning stopped, unbound, unplugged, ...), as well as in the process of being activated or deactivated, i.e. between the two states (these states are called "activating", "deactivating"). A special "failed" state is available as well, which is very similar to "inactive" and is entered when the service failed in some way (process returned error code on exit, or crashed, an operation timed out, or after too many restarts). If this state is entered, the cause will be logged, for later reference. Note that the various unit types may have a number of additional substates, which are mapped to the five generalized unit states described here.

The following unit types are available:

1. Service units, which start and control daemons and the processes they consist of. For details, see **systemd.service(5)**.
2. Socket units, which encapsulate local IPC or network sockets in the system, useful for socket-based activation. For details about socket units, see **systemd.socket(5)**, for details on socket-based activation and other forms of activation, see **daemon(7)**.
3. Target units are useful to group units, or provide well-known synchronization points during boot-up, see **systemd.target(5)**.
4. Device units expose kernel devices in systemd and may be used to implement device-based activation. For details, see **systemd.device(5)**.
5. Mount units control mount points in the file system, for details see **systemd.mount(5)**.
6. Automount units provide automount capabilities, for on-demand mounting of file systems as well as parallelized boot-up. See **systemd.automount(5)**.
7. Timer units are useful for triggering activation of other units based on timers. You may find details in **systemd.timer(5)**.
8. Swap units are very similar to mount units and encapsulate memory swap partitions or files of the operating system. They are described in **systemd.swap(5)**.

9. Path units may be used to activate other services when file system objects change or are modified. See **systemd.path(5)**.
10. Slice units may be used to group units which manage system processes (such as service and scope units) in a hierarchical tree for resource management purposes. See **systemd.slice(5)**.
11. Scope units are similar to service units, but manage foreign processes instead of starting them as well. See **systemd.scope(5)**.

Units are named as their configuration files. Some units have special semantics. A detailed list is available in **systemd.special(7)**.

systemd knows various kinds of dependencies, including positive and negative requirement dependencies (i.e. *Requires=* and *Conflicts=*) as well as ordering dependencies (*After=* and *Before=*). NB: ordering and requirement dependencies are orthogonal. If only a requirement dependency exists between two units (e.g. `foo.service` requires `bar.service`), but no ordering dependency (e.g. `foo.service` after `bar.service`) and both are requested to start, they will be started in parallel. It is a common pattern that both requirement and ordering dependencies are placed between two units. Also note that the majority of dependencies are implicitly created and maintained by systemd. In most cases, it should be unnecessary to declare additional dependencies manually, however it is possible to do this.

Application programs and units (via dependencies) may request state changes of units. In systemd, these requests are encapsulated as 'jobs' and maintained in a job queue. Jobs may succeed or can fail, their execution is ordered based on the ordering dependencies of the units they have been scheduled for.

On boot systemd activates the target unit `default.target` whose job is to activate on-boot services and other on-boot units by pulling them in via dependencies. Usually, the unit name is just an alias (symlink) for either `graphical.target` (for fully-featured boots into the UI) or `multi-user.target` (for limited console-only boots for use in embedded or server environments, or similar; a subset of `graphical.target`). However, it is at the discretion of the administrator to configure it as an alias to any other target unit. See **systemd.special(7)** for details about these target units.

systemd only keeps a minimal set of units loaded into memory. Specifically, the only units that are kept loaded into memory are those for which at least one of the following conditions is true:

1. It is in an active, activating, deactivating or failed state (i.e. in any unit state except for "inactive")
2. It has a job queued for it
3. It is a dependency of at least one other unit that is loaded into memory
4. It has some form of resource still allocated (e.g. a service unit that is inactive but for which a process is still lingering that ignored the request to be terminated)
5. It has been pinned into memory programmatically by a D-Bus call

systemd will automatically and implicitly load units from disk — if they are not loaded yet — as soon as operations are requested for them. Thus, in many respects, the fact whether a unit is loaded or not is invisible to clients. Use **systemctl list-units --all** to comprehensively list all units currently loaded. Any unit for which none of the conditions above applies is promptly unloaded. Note that when a unit is unloaded from memory its accounting data is flushed out too. However, this data is generally not lost, as a journal log record is generated declaring the consumed resources whenever a unit shuts down.

Processes systemd spawns are placed in individual Linux control groups named after the unit which they belong to in the private systemd hierarchy. (see **cgroups.txt**^[1] for more information about control groups, or short "cgroups"). systemd uses this to effectively keep track of processes. Control group information is maintained in the kernel, and is accessible via the file system hierarchy (beneath `/sys/fs/cgroup/systemd/`), or in tools such as **systemd-cgls(1)** or **ps(1)** (**ps xawf -eo pid,user,cgroup,args** is particularly useful to list all processes and the systemd units they belong to.).

systemd is compatible with the SysV init system to a large degree: SysV init scripts are supported and simply read as an alternative (though limited) configuration file format. The SysV `/dev/initctl` interface is provided, and compatibility implementations of the various SysV client tools are available. In addition to

that, various established Unix functionality such as /etc/fstab or the utmp database are supported.

systemd has a minimal transaction system: if a unit is requested to start up or shut down it will add it and all its dependencies to a temporary transaction. Then, it will verify if the transaction is consistent (i.e. whether the ordering of all units is cycle-free). If it is not, systemd will try to fix it up, and removes non-essential jobs from the transaction that might remove the loop. Also, systemd tries to suppress non-essential jobs in the transaction that would stop a running service. Finally it is checked whether the jobs of the transaction contradict jobs that have already been queued, and optionally the transaction is aborted then. If all worked out and the transaction is consistent and minimized in its impact it is merged with all already outstanding jobs and added to the run queue. Effectively this means that before executing a requested operation, systemd will verify that it makes sense, fixing it if possible, and only failing if it really cannot work.

Note that transactions are generated independently of a unit's state at runtime, hence, for example, if a start job is requested on an already started unit, it will still generate a transaction and wake up any inactive dependencies (and cause propagation of other jobs as per the defined relationships). This is because the enqueued job is at the time of execution compared to the target unit's state and is marked successful and complete when both satisfy. However, this job also pulls in other dependencies due to the defined relationships and thus leads to, in our example, start jobs for any of those inactive units getting queued as well.

systemd contains native implementations of various tasks that need to be executed as part of the boot process. For example, it sets the hostname or configures the loopback network device. It also sets up and mounts various API file systems, such as /sys/ or /proc/.

For more information about the concepts and ideas behind systemd, please refer to the [Original Design Document](#)^[2].

Note that some but not all interfaces provided by systemd are covered by the [Interface Portability and Stability Promise](#)^[3].

Units may be generated dynamically at boot and system manager reload time, for example based on other configuration files or parameters passed on the kernel command line. For details, see [systemd.generator\(7\)](#).

The D-Bus API of [systemd](#) is described in [org.freedesktop.systemd1\(5\)](#) and [org.freedesktop.LogControl1\(5\)](#).

Systems which invoke systemd in a container or initrd environment should implement the [Container Interface](#)^[4] or [initrd Interface](#)^[5] specifications, respectively.

DIRECTORIES

System unit directories

The systemd system manager reads unit configuration from various directories. Packages that want to install unit files shall place them in the directory returned by [pkg-config systemd](#)

[--variable=systemdsystemunitdir](#). Other directories checked are /usr/local/lib/systemd/system and /lib/systemd/system. User configuration always takes precedence. [pkg-config systemd](#)

[--variable=systemdsystemconfdir](#) returns the path of the system configuration directory. Packages should alter the content of these directories only with the [enable](#) and [disable](#) commands of the [systemctl\(1\)](#) tool. Full list of directories is provided in [systemd.unit\(5\)](#).

User unit directories

Similar rules apply for the user unit directories. However, here the [XDG Base Directory specification](#)^[6] is followed to find units. Applications should place their unit files in the directory returned by [pkg-config systemd](#) [--variable=systemduserunitdir](#). Global configuration is done in the directory reported by [pkg-config systemd](#) [--variable=systemduserconfdir](#). The [enable](#) and [disable](#) commands of the [systemctl\(1\)](#) tool can handle both global (i.e. for all users) and private (for one user) enabling/disabling of units. Full list of directories is provided in [systemd.unit\(5\)](#).

SysV init scripts directory

The location of the SysV init script directory varies between distributions. If systemd cannot find a native unit file for a requested service, it will look for a SysV init script of the same name (with the

.service suffix removed).

SysV runlevel link farm directory

The location of the SysV runlevel link farm directory varies between distributions. systemd will take the link farm into account when figuring out whether a service shall be enabled. Note that a service unit with a native unit configuration file cannot be started by activating it in the SysV runlevel link farm.

SIGNALS

SIGTERM

Upon receiving this signal the systemd system manager serializes its state, reexecutes itself and deserializes the saved state again. This is mostly equivalent to **systemctl daemon-reexec**.

systemd user managers will start the exit.target unit when this signal is received. This is mostly equivalent to **systemctl --user start exit.target --job-mode=replace-irreversibly**.

SIGINT

Upon receiving this signal the systemd system manager will start the ctrl-alt-del.target unit. This is mostly equivalent to **systemctl start ctrl-alt-del.target --job-mode=replace-irreversibly**. If this signal is received more than 7 times per 2s, an immediate reboot is triggered. Note that pressing Ctrl+Alt+Del on the console will trigger this signal. Hence, if a reboot is hanging, pressing Ctrl+Alt+Del more than 7 times in 2 seconds is a relatively safe way to trigger an immediate reboot.

systemd user managers treat this signal the same way as **SIGTERM**.

SIGWINCH

When this signal is received the systemd system manager will start the kbrequest.target unit. This is mostly equivalent to **systemctl start kbrequest.target**.

This signal is ignored by systemd user managers.

SIGPWR

When this signal is received the systemd manager will start the sigpwr.target unit. This is mostly equivalent to **systemctl start sigpwr.target**.

SIGUSR1

When this signal is received the systemd manager will try to reconnect to the D-Bus bus.

SIGUSR2

When this signal is received the systemd manager will log its complete state in human-readable form. The data logged is the same as printed by **systemd-analyze dump**.

SIGHUP

Reloads the complete daemon configuration. This is mostly equivalent to **systemctl daemon-reload**.

SIGRTMIN+0

Enters default mode, starts the default.target unit. This is mostly equivalent to **systemctl isolate default.target**.

SIGRTMIN+1

Enters rescue mode, starts the rescue.target unit. This is mostly equivalent to **systemctl isolate rescue.target**.

SIGRTMIN+2

Enters emergency mode, starts the emergency.service unit. This is mostly equivalent to **systemctl isolate emergency.service**.

SIGRTMIN+3

Halts the machine, starts the halt.target unit. This is mostly equivalent to **systemctl start halt.target --job-mode=replace-irreversibly**.

SIGRTMIN+4

Powers off the machine, starts the poweroff.target unit. This is mostly equivalent to **systemctl start poweroff.target --job-mode=replace-irreversibly**.

SIGRTMIN+5

Reboots the machine, starts the reboot.target unit. This is mostly equivalent to **systemctl start reboot.target --job-mode=replace-irreversibly**.

SIGRTMIN+6

Reboots the machine via kexec, starts the kexec.target unit. This is mostly equivalent to **systemctl start kexec.target --job-mode=replace-irreversibly**.

SIGRTMIN+13

Immediately halts the machine.

SIGRTMIN+14

Immediately powers off the machine.

SIGRTMIN+15

Immediately reboots the machine.

SIGRTMIN+16

Immediately reboots the machine with kexec.

SIGRTMIN+20

Enables display of status messages on the console, as controlled via *systemd.show_status=1* on the kernel command line.

SIGRTMIN+21

Disables display of status messages on the console, as controlled via *systemd.show_status=0* on the kernel command line.

SIGRTMIN+22

Sets the service manager's log level to "debug", in a fashion equivalent to *systemd.log_level=debug* on the kernel command line.

SIGRTMIN+23

Restores the log level to its configured value. The configured value is derived from – in order of priority – the value specified with *systemd.log_level=* on the kernel command line, or the value specified with **LogLevel=** in the configuration file, or the built-in default of "info".

SIGRTMIN+24

Immediately exits the manager (only available for --user instances).

SIGRTMIN+26

Restores the log target to its configured value. The configured value is derived from – in order of priority – the value specified with *systemd.log_target=* on the kernel command line, or the value specified with **LogTarget=** in the configuration file, or the built-in default.

SIGRTMIN+27, SIGRTMIN+28

Sets the log target to "console" on **SIGRTMIN+27** (or "kmsg" on **SIGRTMIN+28**), in a fashion equivalent to *systemd.log_target=console* (or *systemd.log_target=kmsg* on **SIGRTMIN+28**) on the kernel command line.

ENVIRONMENT

The environment block for the system manager is initially set by the kernel. (In particular, "key=value" assignments on the kernel command line are returned into environment variables for PID 1). For the user manager, the system manager sets the environment as described in the "Environment Variables in Spawning Processes" section of **systemd.exec(5)**. The *DefaultEnvironment=* setting in the system manager applies to all services including *user@.service*. Additional entries may be configured (as for any other service) through the *Environment=* and *EnvironmentFile=* settings for *user@.service* (see **systemd.exec(5)**). Also, additional environment variables may be set through the *ManagerEnvironment=* setting in **systemd-system.conf(5)** and **systemd-user.conf(5)**.

Some of the variables understood by **systemd**:

\$SYSTEMD_LOG_LEVEL

The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog(3)** for more information.

This can be overridden with **--log-level=**.

\$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This can be overridden with **--log-color=**.

\$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This can be overridden with **--log-time=**.

\$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

This can be overridden with **--log-location=**.

\$SYSTEMD_LOG_TID

A boolean. If true, messages will be prefixed with the current numerical thread ID (TID).

\$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console-prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see **syslog(3)**), **kmsg** (log to the kernel circular log buffer), **journal** (log to the journal), **journal-or-kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

This can be overridden with **--log-target=**.

\$XDG_CONFIG_HOME*, *\$XDG_CONFIG_DIRS*, *\$XDG_DATA_HOME*, *\$XDG_DATA_DIRS

The systemd user manager uses these variables in accordance to the [XDG Base Directory specification](#)^[6] to find its configuration.

***\$SYSTEMD_UNIT_PATH*, *\$SYSTEMD_GENERATOR_PATH*,**

\$SYSTEMD_ENVIRONMENT_GENERATOR_PATH

Controls where systemd looks for unit files and generators.

These variables may contain a list of paths, separated by colons (:). When set, if the list ends with an empty component ("::"), this list is prepended to the usual set of paths. Otherwise, the specified list replaces the usual set of paths.

\$SYSTEMD_PAGER

Pager to use when **--no-pager** is not given; overrides **\$PAGER**. If neither **\$SYSTEMD_PAGER** nor **\$PAGER** are set, a set of well-known pager implementations are tried in turn, including **less(1)** and **more(1)**, until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **--no-pager**.

\$SYSTEMD_LESS

Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K

This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of `$SYSTEMD_LESS` does not include "K", and the pager that is invoked is **less**, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X

This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See **less(1)** for more discussion.

`$SYSTEMD_LESSCHARSET`

Override the charset passed to **less** (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

`$SYSTEMD_PAGERSECURE`

Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If `$SYSTEMD_PAGERSECURE` is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see `geteuid(2)` and `sd_pid_get_owner_uid(3)`. In secure mode, `LESSSECURE=1` will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When `$SYSTEMD_PAGERSECURE` is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only **less(1)** implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under **sudo(8)** or **pkexec(1)**, care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as described above. Setting `SYSTEMD_PAGERSECURE=0` or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the `$SYSTEMD_PAGER` or `$PAGER` variables are to be honoured, `$SYSTEMD_PAGERSECURE` must be set too. It might be reasonable to completely disable the pager using `--no-pager` instead.

`$SYSTEMD_COLORS`

Takes a boolean argument. When true, **systemd** and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on `$TERM` and what the console is connected to.

`$SYSTEMD_URLIFY`

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that **systemd** makes based on `$TERM` and other conditions.

`$LISTEN_PID`, `$LISTEN_FDS`, `$LISTEN_FDNAME`

Set by **systemd** for supervised processes during socket-based activation. See `sd_listen_fds(3)` for more information.

`$NOTIFY_SOCKET`

Set by **systemd** for supervised processes for status and start-up completion notification. See `sd_notify(3)` for more information.

For further environment variables understood by **systemd** and its various components, see [Known Environment Variables](#)^[7].

KERNEL COMMAND LINE

When run as the system instance systemd parses a number of options listed below. They can be specified as kernel command line arguments^[8], or through the "SystemdOptions" EFI variable (on EFI systems). The kernel command line has higher priority. Following variables are understood:

systemd.unit=, *rd.systemd.unit=*

Overrides the unit to activate on boot. Defaults to default.target. This may be used to temporarily boot into a different boot unit, for example rescue.target or emergency.service. See **systemd.special(7)** for details about these units. The option prefixed with "rd." is honored only in the initial RAM disk (initrd), while the one that is not prefixed only in the main system.

systemd.dump_core

Takes a boolean argument or enables the option if specified without an argument. If enabled, the systemd manager (PID 1) dumps core when it crashes. Otherwise, no core dump is created. Defaults to enabled.

systemd.crash_chvt

Takes a positive integer, or a boolean argument. Can be also specified without an argument, with the same effect as a positive boolean. If a positive integer (in the range 1–63) is specified, the system manager (PID 1) will activate the specified virtual terminal when it crashes. Defaults to disabled, meaning that no such switch is attempted. If set to enabled, the virtual terminal the kernel messages are written to is used instead.

systemd.crash_shell

Takes a boolean argument or enables the option if specified without an argument. If enabled, the system manager (PID 1) spawns a shell when it crashes, after a 10s delay. Otherwise, no shell is spawned. Defaults to disabled, for security reasons, as the shell is not protected by password authentication.

systemd.crash_reboot

Takes a boolean argument or enables the option if specified without an argument. If enabled, the system manager (PID 1) will reboot the machine automatically when it crashes, after a 10s delay. Otherwise, the system will hang indefinitely. Defaults to disabled, in order to avoid a reboot loop. If combined with *systemd.crash_shell*, the system is rebooted after the shell exits.

systemd.confirm_spawn

Takes a boolean argument or a path to the virtual console where the confirmation messages should be emitted. Can be also specified without an argument, with the same effect as a positive boolean. If enabled, the system manager (PID 1) asks for confirmation when spawning processes using **/dev/console**. If a path or a console name (such as "ttyS0") is provided, the virtual console pointed to by this path or described by the give name will be used instead. Defaults to disabled.

systemd.service_watchdogs=

Takes a boolean argument. If disabled, all service runtime watchdogs (**WatchdogSec=**) and emergency actions (e.g. **OnFailure=** or **StartLimitAction=**) are ignored by the system manager (PID 1); see **systemd.service(5)**. Defaults to enabled, i.e. watchdogs and failure actions are processed normally. The hardware watchdog is not affected by this option.

systemd.show_status

Takes a boolean argument or the constants **error** and **auto**. Can be also specified without an argument, with the same effect as a positive boolean. If enabled, the system manager (PID 1) shows terse service status updates on the console during bootup. With **error**, only messages about failures are shown, but boot is otherwise quiet. **auto** behaves like **false** until there is a significant delay in boot. Defaults to enabled, unless **quiet** is passed as kernel command line option, in which case it defaults to **error**. If specified overrides the system manager configuration file option **ShowStatus=**, see **systemd-system.conf(5)**.

systemd.status_unit_format=

Takes **name**, **description** or **combined** as the value. If **name**, the system manager will use unit names in status messages. If **combined**, the system manager will use unit names and description in status

messages. When specified, overrides the system manager configuration file option `StatusUnitFormat=`, see [systemd-system.conf\(5\)](#).

`systemd.log_color`, `systemd.log_level=`, `systemd.log_location`, `systemd.log_target=`, `systemd.log_time`, `systemd.log_tid`

Controls log output, with the same effect as the `$SYSTEMD_LOG_COLOR`, `$SYSTEMD_LOG_LEVEL`, `$SYSTEMD_LOG_LOCATION`, `$SYSTEMD_LOG_TARGET`, `$SYSTEMD_LOG_TIME`, and `$SYSTEMD_LOG_TID` environment variables described above. `systemd.log_color`, `systemd.log_location`, `systemd.log_time`, and `systemd.log_tid=` can be specified without an argument, with the same effect as a positive boolean.

`systemd.default_standard_output=`, `systemd.default_standard_error=`

Controls default standard output and error output for services and sockets. That is, controls the default for `StandardOutput=` and `StandardError=` (see [systemd.exec\(5\)](#) for details). Takes one of `inherit`, `null`, `tty`, `journal`, `journal+console`, `kmsg`, `kmsg+console`. If the argument is omitted `systemd.default-standard-output=` defaults to `journal` and `systemd.default-standard-error=` to `inherit`.

`systemd.setenv=`

Takes a string argument in the form `VARIABLE=VALUE`. May be used to set default environment variables to add to forked child processes. May be used more than once to set multiple variables.

`systemd.machine_id=`

Takes a 32 character hex value to be used for setting the machine-id. Intended mostly for network booting where the same machine-id is desired for every boot.

`systemd.unified_cgroup_hierarchy`

When specified without an argument or with a true argument, enables the usage of [unified cgroup hierarchy](#)^[9] (a.k.a. cgroups-v2). When specified with a false argument, fall back to hybrid or full legacy cgroup hierarchy.

If this option is not specified, the default behaviour is determined during compilation (the `-Ddefault-hierarchy=` meson option). If the kernel does not support unified cgroup hierarchy, the legacy hierarchy will be used even if this option is specified.

`systemd.legacy_systemd_cgroup_controller`

Takes effect if the full unified cgroup hierarchy is not used (see previous option). When specified without an argument or with a true argument, disables the use of "hybrid" cgroup hierarchy (i.e. a cgroups-v2 tree used for systemd, and [legacy cgroup hierarchy](#)^[10], a.k.a. cgroups-v1, for other controllers), and forces a full "legacy" mode. When specified with a false argument, enables the use of "hybrid" hierarchy.

If this option is not specified, the default behaviour is determined during compilation (the `-Ddefault-hierarchy=` meson option). If the kernel does not support unified cgroup hierarchy, the legacy hierarchy will be used even if this option is specified.

`quiet`

Turn off status output at boot, much like `systemd.show_status=no` would. Note that this option is also read by the kernel itself and disables kernel log output. Passing this option hence turns off the usual output from both the system manager and the kernel.

`debug`

Turn on debugging output. This is equivalent to `systemd.log_level=debug`. Note that this option is also read by the kernel itself and enables kernel debug output. Passing this option hence turns on the debug output from both the system manager and the kernel.

`emergency, rd.emergency, -b`

Boot into emergency mode. This is equivalent to `systemd.unit=emergency.target` or `rd.systemd.unit=emergency.target`, respectively, and provided for compatibility reasons and to be easier to type.

rescue, rd.rescue, single, s, S, 1

Boot into rescue mode. This is equivalent to `systemd.unit=rescue.target` or `rd.systemd.unit=rescue.target`, respectively, and provided for compatibility reasons and to be easier to type.

2, 3, 4, 5

Boot into the specified legacy SysV runlevel. These are equivalent to `systemd.unit=runlevel2.target`, `systemd.unit=runlevel3.target`, `systemd.unit=runlevel4.target`, and `systemd.unit=runlevel5.target`, respectively, and provided for compatibility reasons and to be easier to type.

locale.LANG=, locale.LANGUAGE=, locale.LC_CTYPE=, locale.LC_NUMERIC=, locale.LC_TIME=, locale.LC_COLLATE=, locale.LC_MONETARY=, locale.LC_MESSAGES=, locale.LC_PAPER=, locale.LC_NAME=, locale.LC_ADDRESS=, locale.LC_TELEPHONE=, locale.LC_MEASUREMENT=, locale.LC_IDENTIFICATION=

Set the system locale to use. This overrides the settings in `/etc/locale.conf`. For more information, see **locale.conf(5)** and **locale(7)**.

For other kernel command line parameters understood by components of the core OS, please refer to **kernel-command-line(7)**.

OPTIONS

systemd is only very rarely invoked directly, since it is started early and is already running by the time users may interact with it. Normally, tools like **systemctl(1)** are used to give commands to the manager. Since **systemd** is usually not invoked directly, the options listed below are mostly useful for debugging and special purposes.

Introspection and debugging options

Those options are used for testing and introspection, and **systemd** may be invoked with them at any time:

--dump-configuration-items

Dump understood unit configuration items. This outputs a terse but complete list of configuration items understood in unit definition files.

--dump-bus-properties

Dump exposed bus properties. This outputs a terse but complete list of properties exposed on D-Bus.

--test

Determine the initial start-up transaction (i.e. the list of jobs enqueued at start-up), dump it and exit — without actually executing any of the determined jobs. This option is useful for debugging only.

Note that during regular service manager start-up additional units not shown by this operation may be started, because hardware, socket, bus or other kinds of activation might add additional jobs as the transaction is executed. Use **--system** to request the initial transaction of the system service manager (this is also the implied default), combine with **--user** to request the initial transaction of the per-user service manager instead.

--system, --user

When used in conjunction with **--test**, selects whether to calculate the initial transaction for the system instance or for a per-user instance. These options have no effect when invoked without **--test**, as during regular (i.e. non---**test**) invocations the service manager will automatically detect whether it shall operate in system or per-user mode, by checking whether the PID it is run as is 1 or not. Note that it is not supported booting and maintaining a system with the service manager running in **--system** mode but with a PID other than 1.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

Options that duplicate kernel command line settings

Those options correspond directly to options listed above in "Kernel Command Line". Both forms may be used equivalently for the system manager, but it is recommended to use the forms listed above in this

context, because they are properly namespaced. When an option is specified both on the kernel command line and as a normal command line argument, the latter has higher precedence.

When **systemd** is used as a user manager, the kernel command line is ignored and only the options described below are understood. Nevertheless, **systemd** is usually started in this mode through the **user@.service(5)** service, which is shared between all users. It may be more convenient to use configuration files to modify settings (see **systemd-user.conf(5)**), or environment variables. See the "Environment" section above for a discussion of how the environment block is set.

--unit=

Set default unit to activate on startup. If not specified, defaults to `default.target`. See `systemd.unit=` above.

--dump-core

Enable core dumping on crash. This switch has no effect when running as user instance. Same as `systemd.dump_core=` above.

--crash-vt=VT

Switch to a specific virtual console (VT) on crash. This switch has no effect when running as user instance. Same as `systemd.crash_chvt=` above (but not the different spelling!).

--crash-shell

Run a shell on crash. This switch has no effect when running as user instance. See `systemd.crash_shell=` above.

--crash-reboot

Automatically reboot the system on crash. This switch has no effect when running as user instance. See `systemd.crash_reboot` above.

--confirm-spawn

Ask for confirmation when spawning processes. This switch has no effect when run as user instance. See `systemd.confirm_spawn` above.

--show-status

Show terse unit status information on the console during boot-up and shutdown. See `systemd.show_status` above.

--log-color

Highlight important log messages. See `systemd.log_color` above.

--log-level=

Set log level. See `systemd.log_level` above.

--log-location

Include code location in log messages. See `systemd.log_location` above.

--log-target=

Set log target. See `systemd.log_target` above.

--log-time=

Prefix console messages with timestamp. See `systemd.log_time` above.

--machine-id=

Override the machine-id set on the hard drive. See `systemd.machine_id=` above.

--service-watchdogs

Globally enable/disable all service watchdog timeouts and emergency actions. See `systemd.service_watchdogs` above.

--default-standard-output=, --default-standard-error=

Sets the default output or error output for all services and sockets, respectively. See `systemd.default_standard_output=` and `systemd.default_standard_error=` above.

SOCKETS AND FIFOs

/run/systemd/notify

Daemon status notification socket. This is an **AF_UNIX** datagram socket and is used to implement the daemon notification logic as implemented by **sd_notify(3)**.

/run/systemd/private

Used internally as communication channel between **systemctl(1)** and the systemd process. This is an **AF_UNIX** stream socket. This interface is private to systemd and should not be used in external projects.

/dev/initctl

Limited compatibility support for the SysV client interface, as implemented by the **systemd-initctl.service** unit. This is a named pipe in the file system. This interface is obsolete and should not be used in new applications.

SEE ALSO

The **systemd Homepage**^[11], **systemd-system.conf(5)**, **locale.conf(5)**, **systemctl(1)**, **journalctl(1)**, **systemd-notify(1)**, **daemon(7)**, **sd-daemon(3)**, **org.freedesktop.systemd1(5)**, **systemd.unit(5)**, **systemd.special(7)**, **pkg-config(1)**, **kernel-command-line(7)**, **bootup(7)**, **systemd.directives(7)**

NOTES

1. cgroups.txt
<https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt>
2. Original Design Document
<http://0pointer.de/blog/projects/systemd.html>
3. Interface Portability and Stability Promise
https://systemd.io/PORABILITY_AND_STABILITY/
4. Container Interface
https://systemd.io/CONTAINER_INTERFACE
5. initrd Interface
https://systemd.io/INITRD_INTERFACE
6. XDG Base Directory specification
<http://standards.freedesktop.org/basedir-spec/basedir-spec-latest.html>
7. Known Environment Variables
<https://systemd.io/ENVIRONMENT>
8. If run inside a Linux container these arguments may be passed as command line arguments to systemd itself, next to any of the command line options listed in the Options section above. If run outside of Linux containers, these arguments are parsed from /proc/cmdline instead.
9. unified cgroup hierarchy
<https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html>
10. legacy cgroup hierarchy
<https://www.kernel.org/doc/Documentation/cgroup-v1/>
11. systemd Homepage
<https://www.freedesktop.org/wiki/Software/systemd/>

NAME

`systemd.kill` – Process killing procedure configuration

SYNOPSIS

`service.service, socket.socket, mount.mount, swap.swap, scope.scope`

DESCRIPTION

Unit configuration files for services, sockets, mount points, swap devices and scopes share a subset of configuration options which define the killing procedure of processes belonging to the unit.

This man page lists the configuration options shared by these five unit types. See **systemd.unit(5)** for the common options shared by all unit configuration files, and **systemd.service(5)**, **systemd.socket(5)**, **systemd.swap(5)**, **systemd.mount(5)** and **systemd.scope(5)** for more information on the configuration file options specific to each unit type.

The kill procedure configuration options are configured in the [Service], [Socket], [Mount] or [Swap] section, depending on the unit type.

OPTIONS

KillMode=

Specifies how processes of this unit shall be killed. One of **control-group**, **mixed**, **process**, **none**.

If set to **control-group**, all remaining processes in the control group of this unit will be killed on unit stop (for services: after the stop command is executed, as configured with *ExecStop*=). If set to **mixed**, the **SIGTERM** signal (see below) is sent to the main process while the subsequent **SIGKILL** signal (see below) is sent to all remaining processes of the unit's control group. If set to **process**, only the main process itself is killed (not recommended!). If set to **none**, no process is killed (strongly recommended against!). In this case, only the stop command will be executed on unit stop, but no process will be killed otherwise. Processes remaining alive after stop are left in their control group and the control group continues to exist after stop unless empty.

Note that it is not recommended to set *KillMode*= to **process** or even **none**, as this allows processes to escape the service manager's lifecycle and resource management, and to remain running even while their service is considered stopped and is assumed to not consume any resources.

Processes will first be terminated via **SIGTERM** (unless the signal to send is changed via *KillSignal*= or *RestartKillSignal*=). Optionally, this is immediately followed by a **SIGHUP** (if enabled with *SendSIGHUP*=). If processes still remain after the main process of a unit has exited or the delay configured via the *TimeoutStopSec*= has passed, the termination request is repeated with the **SIGKILL** signal or the signal specified via *FinalKillSignal*= (unless this is disabled via the *SendSIGKILL*= option). See **kill(2)** for more information.

Defaults to **control-group**.

KillSignal=

Specifies which signal to use when stopping a service. This controls the signal that is sent as first step of shutting down a unit (see above), and is usually followed by **SIGKILL** (see above and below). For a list of valid signals, see **signal(7)**. Defaults to **SIGTERM**.

Note that, right after sending the signal specified in this setting, systemd will always send **SIGCONT**, to ensure that even suspended tasks can be terminated cleanly.

RestartKillSignal=

Specifies which signal to use when restarting a service. The same as *KillSignal*= described above, with the exception that this setting is used in a restart job. Not set by default, and the value of *KillSignal*= is used.

SendSIGHUP=

Specifies whether to send **SIGHUP** to remaining processes immediately after sending the signal

configured with *KillSignal*=. This is useful to indicate to shells and shell-like programs that their connection has been severed. Takes a boolean value. Defaults to "no".

SendSIGKILL=

Specifies whether to send **SIGKILL** (or the signal specified by *FinalKillSignal*=) to remaining processes after a timeout, if the normal shutdown procedure left processes of the service around. When disabled, a *KillMode*= of **control-group** or **mixed** service will not restart if processes from prior services exist within the control group. Takes a boolean value. Defaults to "yes".

FinalKillSignal=

Specifies which signal to send to remaining processes after a timeout if *SendSIGKILL*= is enabled. The signal configured here should be one that is not typically caught and processed by services (**SIGTERM** is not suitable). Developers can find it useful to use this to generate a core dump to troubleshoot why a service did not terminate upon receiving the initial **SIGTERM** signal. This can be achieved by configuring *LimitCORE*= and setting *FinalKillSignal*= to either **SIGQUIT** or **SIGABRT**. Defaults to **SIGKILL**.

WatchdogSignal=

Specifies which signal to use to terminate the service when the watchdog timeout expires (enabled through *WatchdogSec*=). Defaults to **SIGABRT**.

SEE ALSO

systemd(1), **systemctl(1)**, **journalctl(1)**, **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, **systemd.swap(5)**, **systemd.mount(5)**, **systemd.exec(5)**, **systemd.directives(7)**, **kill(2)**, **signal(7)**

NAME

`systemd.mount` – Mount unit configuration

SYNOPSIS

`mount.mount`

DESCRIPTION

A unit configuration file whose name ends in ".mount" encodes information about a file system mount point controlled and supervised by systemd.

This man page lists the configuration options specific to this unit type. See **systemd.unit(5)** for the common options of all unit configuration files. The common configuration items are configured in the generic [Unit] and [Install] sections. The mount specific configuration options are configured in the [Mount] section.

Additional options are listed in **systemd.exec(5)**, which define the execution environment the **mount(8)** program is executed in, and in **systemd.kill(5)**, which define the way the processes are terminated, and in **systemd.resource-control(5)**, which configure resource control settings for the processes of the service.

Note that the options *User=* and *Group=* are not useful for mount units. systemd passes two parameters to **mount(8)**; the values of *What=* and *Where=*. When invoked in this way, **mount(8)** does not read any options from /etc/fstab, and must be run as UID 0.

Mount units must be named after the mount point directories they control. Example: the mount point /home/lennart must be configured in a unit file `home-lennart.mount`. For details about the escaping logic used to convert a file system path to a unit name, see **systemd.unit(5)**. Note that mount units cannot be templated, nor is possible to add multiple names to a mount unit by creating additional symlinks to it.

Optionally, a mount unit may be accompanied by an automount unit, to allow on-demand or parallelized mounting. See **systemd.automount(5)**.

Mount points created at runtime (independently of unit files or /etc/fstab) will be monitored by systemd and appear like any other mount unit in systemd. See /proc/self/mountinfo description in **proc(5)**.

Some file systems have special semantics as API file systems for kernel-to-userspace and userspace-to-userspace interfaces. Some of them may not be changed via mount units, and cannot be disabled. For a longer discussion see [API File Systems^{\[1\]}](#).

The **systemd-mount(1)** command allows creating .mount and .automount units dynamically and transiently from the command line.

AUTOMATIC DEPENDENCIES**Implicit Dependencies**

The following dependencies are implicitly added:

- If a mount unit is beneath another mount unit in the file system hierarchy, both a requirement dependency and an ordering dependency between both units are created automatically.
- Block device backed file systems automatically gain *BindsTo=* and *After=* type dependencies on the device unit encapsulating the block device (see below).
- If traditional file system quota is enabled for a mount unit, automatic *Wants=* and *Before=* dependencies on `systemd-quotacheck.service` and `quotaon.service` are added.
- Additional implicit dependencies may be added as result of execution and resource control parameters as documented in **systemd.exec(5)** and **systemd.resource-control(5)**.

Default Dependencies

The following dependencies are added unless *DefaultDependencies=no* is set:

- All mount units acquire automatic *Before=* and *Conflicts=* on `umount.target` in order to be stopped during shutdown.
- Mount units referring to local file systems automatically gain an *After=* dependency on `local-fs-pre.target`, and a *Before=* dependency on `local-fs.target` unless **nofail** mount option is set.

- Network mount units automatically acquire *After=* dependencies on `remote-fs-pre.target`, `network.target` and `network-online.target`, and gain a *Before=* dependency on `remote-fs.target` unless `nofail` mount option is set. Towards the latter a *Wants=* unit is added as well.

Mount units referring to local and network file systems are distinguished by their file system type specification. In some cases this is not sufficient (for example network block device based mounts, such as iSCSI), in which case `_netdev` may be added to the mount option string of the unit, which forces systemd to consider the mount unit a network mount.

FSTAB

Mount units may either be configured via unit files, or via /etc/fstab (see **fstab(5)** for details). Mounts listed in /etc/fstab will be converted into native units dynamically at boot and when the configuration of the system manager is reloaded. In general, configuring mount points through /etc/fstab is the preferred approach. See **systemd-fstab-generator(8)** for details about the conversion.

The NFS mount option `bg` for NFS background mounts as documented in **nfs(5)** is detected by **systemd-fstab-generator** and the options are transformed so that systemd fulfills the job-control implications of that option. Specifically **systemd-fstab-generator** acts as though "x-systemd.mount-timeout=infinity,retry=10000" was prepended to the option list, and "fg,nofail" was appended. Depending on specific requirements, it may be appropriate to provide some of these options explicitly, or to make use of the "x-systemd.automount" option described below instead of using "bg".

When reading /etc/fstab a few special mount options are understood by systemd which influence how dependencies are created for mount points. systemd will create a dependency of type *Wants=* or *Requires=* (see option `nofail` below), from either `local-fs.target` or `remote-fs.target`, depending whether the file system is local or remote.

x–systemd.requires=

Configures a *Requires=* and an *After=* dependency between the created mount unit and another systemd unit, such as a device or mount unit. The argument should be a unit name, or an absolute path to a device node or mount point. This option may be specified more than once. This option is particularly useful for mount point declarations that need an additional device to be around (such as an external journal device for journal file systems) or an additional mount to be in place (such as an overlay file system that merges multiple mount points). See *After=* and *Requires=* in **systemd.unit(5)** for details.

Note that this option always applies to the created mount unit only regardless whether **x–systemd.automount** has been specified.

x–systemd.before=, x–systemd.after=

In the created mount unit, configures a *Before=* or *After=* dependency on another systemd unit, such as a mount unit. The argument should be a unit name or an absolute path to a mount point. This option may be specified more than once. This option is particularly useful for mount point declarations with `nofail` option that are mounted asynchronously but need to be mounted before or after some unit start, for example, before `local-fs.target` unit. See *Before=* and *After=* in **systemd.unit(5)** for details.

Note that these options always apply to the created mount unit only regardless whether **x–systemd.automount** has been specified.

x–systemd.wanted-by=, x–systemd.required-by=

In the created mount unit, configures a *WantedBy=* or *RequiredBy=* dependency on another unit. This option may be specified more than once. If this is specified, the normal automatic dependencies on the created mount unit, e.g., `local-fs.target`, are not automatically created. See *WantedBy=* and *RequiredBy=* in **systemd.unit(5)** for details.

x–systemd.requires-mounts-for=

Configures a *RequiresMountsFor=* dependency between the created mount unit and other mount units. The argument must be an absolute path. This option may be specified more than once. See *RequiresMountsFor=* in **systemd.unit(5)** for details.

x–systemd.device–bound

The block device backed file system will be upgraded to *BindsTo=* dependency. This option is only useful when mounting file systems manually with **mount(8)** as the default dependency in this case is *Requires=*. This option is already implied by entries in /etc/fstab or by mount units.

x–systemd.automount

An automount unit will be created for the file system. See **systemd.automount(5)** for details.

x–systemd.idle–timeout=

Configures the idle timeout of the automount unit. See *TimeoutIdleSec=* in **systemd.automount(5)** for details.

x–systemd.device–timeout=

Configure how long systemd should wait for a device to show up before giving up on an entry from /etc/fstab. Specify a time in seconds or explicitly append a unit such as "s", "min", "h", "ms".

Note that this option can only be used in /etc/fstab, and will be ignored when part of the *Options=* setting in a unit file.

x–systemd.mount–timeout=

Configure how long systemd should wait for the mount command to finish before giving up on an entry from /etc/fstab. Specify a time in seconds or explicitly append a unit such as "s", "min", "h", "ms".

Note that this option can only be used in /etc/fstab, and will be ignored when part of the *Options=* setting in a unit file.

See *TimeoutSec=* below for details.

x–systemd.makefs

The file system will be initialized on the device. If the device is not "empty", i.e. it contains any signature, the operation will be skipped. It is hence expected that this option remains set even after the device has been initialized.

Note that this option can only be used in /etc/fstab, and will be ignored when part of the *Options=* setting in a unit file.

See **systemd-makefs@.service(8)**.

wipefs(8) may be used to remove any signatures from a block device to force **x–systemd.makefs** to reinitialize the device.

x–systemd.growfs

The file system will be grown to occupy the full block device. If the file system is already at maximum size, no action will be performed. It is hence expected that this option remains set even after the file system has been grown. Only certain file system types are supported, see **systemd-makefs@.service(8)** for details.

Note that this option can only be used in /etc/fstab, and will be ignored when part of the *Options=* setting in a unit file.

x–systemd.rw–only

If a mount operation fails to mount the file system read–write, it normally tries mounting the file system read–only instead. This option disables that behaviour, and causes the mount to fail immediately instead. This option is translated into the *ReadWriteOnly=* setting in a unit file.

_netdev

Normally the file system type is used to determine if a mount is a "network mount", i.e. if it should only be started after the network is available. Using this option overrides this detection and specifies

that the mount requires network.

Network mount units are ordered between `remote-fs-pre.target` and `remote-fs.target`, instead of `local-fs-pre.target` and `local-fs.target`. They also pull in `network-online.target` and are ordered after it and `network.target`.

noauto, auto

With **noauto**, the mount unit will not be added as a dependency for `local-fs.target` or `remote-fs.target`. This means that it will not be mounted automatically during boot, unless it is pulled in by some other unit. The **auto** option has the opposite meaning and is the default.

Note that if **x-systemd.automount** (see above) is used, neither **auto** nor **noauto** have any effect. The matching automount unit will be added as a dependency to the appropriate target.

nofail

With **nofail**, this mount will be only wanted, not required, by `local-fs.target` or `remote-fs.target`. Moreover the mount unit is not ordered before these target units. This means that the boot will continue without waiting for the mount unit and regardless whether the mount point can be mounted successfully.

x-initrd.mount

An additional filesystem to be mounted in the initramfs. See `initrd-fs.target` description in **systemd.special(7)**.

If a mount point is configured in both `/etc/fstab` and a unit file that is stored below `/usr/`, the former will take precedence. If the unit file is stored below `/etc/`, it will take precedence. This means: native unit files take precedence over traditional configuration files, but this is superseded by the rule that configuration in `/etc/` will always take precedence over configuration in `/usr/`.

OPTIONS

Mount files must include a [Mount] section, which carries information about the file system mount points it supervises. A number of options that may be used in this section are shared with other unit types. These options are documented in **systemd.exec(5)** and **systemd.kill(5)**. The options specific to the [Mount] section of mount units are the following:

What=

Takes an absolute path of a device node, file or other resource to mount. See **mount(8)** for details. If this refers to a device node, a dependency on the respective device unit is automatically created. (See **systemd.device(5)** for more information.) This option is mandatory. Note that the usual specifier expansion is applied to this setting, literal percent characters should hence be written as "%%". If this mount is a bind mount and the specified path does not exist yet it is created as directory.

Where=

Takes an absolute path of a file or directory for the mount point; in particular, the destination cannot be a symbolic link. If the mount point does not exist at the time of mounting, it is created as directory. This string must be reflected in the unit filename. (See above.) This option is mandatory.

Type=

Takes a string for the file system type. See **mount(8)** for details. This setting is optional.

Options=

Mount options to use when mounting. This takes a comma-separated list of options. This setting is optional. Note that the usual specifier expansion is applied to this setting, literal percent characters should hence be written as "%%".

SloppyOptions=

Takes a boolean argument. If true, parsing of the options specified in *Options=* is relaxed, and unknown mount options are tolerated. This corresponds with **mount(8)**'s `-s` switch. Defaults to off.

LazyUnmount=

Takes a boolean argument. If true, detach the filesystem from the filesystem hierarchy at time of the

unmount operation, and clean up all references to the filesystem as soon as they are not busy anymore. This corresponds with **umount(8)**'s *-l* switch. Defaults to off.

ReadOnly=

Takes a boolean argument. If false, a mount point that shall be mounted read-write but cannot be mounted so is retried to be mounted read-only. If true the operation will fail immediately after the read-write mount attempt did not succeed. This corresponds with **mount(8)**'s *-w* switch. Defaults to off.

ForceUnmount=

Takes a boolean argument. If true, force an unmount (in case of an unreachable NFS system). This corresponds with **umount(8)**'s *-f* switch. Defaults to off.

DirectoryMode=

Directories of mount points (and any parent directories) are automatically created if needed. This option specifies the file system access mode used when creating these directories. Takes an access mode in octal notation. Defaults to 0755.

TimeoutSec=

Configures the time to wait for the mount command to finish. If a command does not exit within the configured time, the mount will be considered failed and be shut down again. All commands still running will be terminated forcibly via **SIGTERM**, and after another delay of this time with **SIGKILL**. (See **KillMode=** in **systemd.kill(5)**.) Takes a unit-less value in seconds, or a time span value such as "5min 20s". Pass 0 to disable the timeout logic. The default value is set from **DefaultTimeoutStartSec=** option in **systemd-system.conf(5)**.

Check **systemd.exec(5)** and **systemd.kill(5)** for more settings.

SEE ALSO

systemd(1), **systemctl(1)**, **systemd-system.conf(5)**, **systemd.unit(5)**, **systemd.exec(5)**, **systemd.kill(5)**, **systemd.resource-control(5)**, **systemd.service(5)**, **systemd.device(5)**, **proc(5)**, **mount(8)**, **systemd-fstab-generator(8)**, **systemd.directives(7)**, **systemd-mount(1)**

NOTES

1. API File Systems

<https://www.freedesktop.org/wiki/Software/systemd/APIFileSystems>

NAME

`systemd.network` – Network configuration

SYNOPSIS

network.network

DESCRIPTION

A plain ini–style text file that encodes network configuration for matching network interfaces, used by **systemd-networkd(8)**. See **systemd.syntax(7)** for a general description of the syntax.

The main network file must have the extension `.network`; other extensions are ignored. Networks are applied to links whenever the links appear.

The `.network` files are read from the files located in the system network directories `/lib/systemd/network` and `/usr/local/lib/systemd/network`, the volatile runtime network directory `/run/systemd/network` and the local administration network directory `/etc/systemd/network`. All configuration files are collectively sorted and processed in lexical order, regardless of the directories in which they live. However, files with identical filenames replace each other. Files in `/etc/` have the highest priority, files in `/run/` take precedence over files with the same name under `/usr/`. This can be used to override a system–supplied configuration file with a local file if needed. As a special case, an empty file (file size 0) or symlink with the same name pointing to `/dev/null` disables the configuration file entirely (it is "masked").

Along with the network file `foo.network`, a "drop–in" directory `foo.network.d/` may exist. All files with the suffix `".conf"` from this directory will be merged in the alphanumeric order and parsed after the main file itself has been parsed. This is useful to alter or add configuration settings, without having to modify the main configuration file. Each drop–in file must have appropriate section headers.

In addition to `/etc/systemd/network`, drop–in `".d"` directories can be placed in `/lib/systemd/network` or `/run/systemd/network` directories. Drop–in files in `/etc/` take precedence over those in `/run/` which in turn take precedence over those in `/lib/`. Drop–in files under any of these directories take precedence over the main network file wherever located.

[MATCH] SECTION OPTIONS

The network file contains a `[Match]` section, which determines if a given network file may be applied to a given device; and a `[Network]` section specifying how the device should be configured. The first (in lexical order) of the network files that matches a given device is applied, all later files are ignored, even if they match as well.

A network file is said to match a network interface if all matches specified by the `[Match]` section are satisfied. When a network file does not contain valid settings in `[Match]` section, then the file will match all interfaces and **systemd–networkd** warns about that. Hint: to avoid the warning and to make it clear that all interfaces shall be matched, add the following:

`Name=*`

The following keys are accepted:

MACAddress=

A whitespace–separated list of hardware addresses. Use full colon–, hyphen– or dot–delimited hexadecimal. See the example below. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list of hardware addresses defined prior to this is reset.

Example:

`MACAddress=01:23:45:67:89:ab 00–11–22–33–44–55 AABB.CCDD.EEFF`

PermanentMACAddress=

A whitespace–separated list of hardware's permanent addresses. While `MACAddress=` matches the device's current MAC address, this matches the device's permanent MAC address, which may be

different from the current one. Use full colon–, hyphen– or dot–delimited hexadecimal. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list of hardware addresses defined prior to this is reset.

Path=

A whitespace-separated list of shell-style globs matching the persistent path, as exposed by the udev property *ID_PATH*.

Driver=

A whitespace-separated list of shell-style globs matching the driver currently bound to the device, as exposed by the udev property *ID_NET_DRIVER* of its parent device, or if that is not set, the driver as exposed by **ethtool -i** of the device itself. If the list is prefixed with a "!", the test is inverted.

Type=

A whitespace-separated list of shell-style globs matching the device type, as exposed by **networkctl list**. If the list is prefixed with a "!", the test is inverted. Some valid values are "ether", "loopback", "wlan", "wwan". Valid types are named either from the udev "DEVTYPE" attribute, or "ARPHRD_" macros in `linux/if_arp.h`, so this is not comprehensive.

Property=

A whitespace-separated list of udev property names with their values after equals sign ("="). If multiple properties are specified, the test results are ANDed. If the list is prefixed with a "!", the test is inverted. If a value contains white spaces, then please quote whole key and value pair. If a value contains quotation, then please escape the quotation with "\\".

Example: if a .link file has the following:

```
Property=ID_MODEL_ID=9999 "ID_VENDOR_FROM_DATABASE=vendor name" "KEY=with \"quotation\""
```

then, the .link file matches only when an interface has all the above three properties.

Name=

A whitespace-separated list of shell-style globs matching the device name, as exposed by the udev property "INTERFACE", or device's alternative names. If the list is prefixed with a "!", the test is inverted.

WLANInterfaceType=

A whitespace-separated list of wireless network type. Supported values are "ad-hoc", "station", "ap", "ap-vlan", "wds", "monitor", "mesh-point", "p2p-client", "p2p-go", "p2p-device", "ocb", and "nan". If the list is prefixed with a "!", the test is inverted.

SSID=

A whitespace-separated list of shell-style globs matching the SSID of the currently connected wireless LAN. If the list is prefixed with a "!", the test is inverted.

BSSID=

A whitespace-separated list of hardware address of the currently connected wireless LAN. Use full colon–, hyphen– or dot–delimited hexadecimal. See the example in *MACAddress*=. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list is reset.

Host=

Matches against the hostname or machine ID of the host. See *ConditionHost*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Virtualization=

Checks whether the system is executed in a virtualized environment and optionally test whether it is a specific implementation. See *ConditionVirtualization*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously

assigned value is cleared.

KernelCommandLine=

Checks whether a specific kernel command line option is set. See *ConditionKernelCommandLine*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

KernelVersion=

Checks whether the kernel version (as reported by **uname -r**) matches a certain expression. See *ConditionKernelVersion*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Architecture=

Checks whether the system is running on a specific architecture. See *ConditionArchitecture*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Firmware=

Checks whether the system is running on a machine with the specified firmware. See *ConditionFirmware*= in **systemd.unit(5)** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

[LINK] SECTION OPTIONS

The [Link] section accepts the following keys:

MACAddress=

The hardware address to set for the device.

MTUBytes=

The maximum transmission unit in bytes to set for the device. The usual suffixes K, M, G, are supported and are understood to the base of 1024.

Note that if IPv6 is enabled on the interface, and the MTU is chosen below 1280 (the minimum MTU for IPv6) it will automatically be increased to this value.

ARP=

Takes a boolean. If set to true, the ARP (low-level Address Resolution Protocol) for this interface is enabled. When unset, the kernel's default will be used.

For example, disabling ARP is useful when creating multiple MACVLAN or VLAN virtual interfaces atop a single lower-level physical interface, which will then only serve as a link/"bridge" device aggregating traffic to the same physical link and not participate in the network otherwise. Defaults to unset.

Multicast=

Takes a boolean. If set to true, the multicast flag on the device is enabled. Defaults to unset.

AllMulticast=

Takes a boolean. If set to true, the driver retrieves all multicast packets from the network. This happens when multicast routing is enabled. Defaults to unset.

Promiscuous=

Takes a boolean. If set to true, promiscuous mode of the interface is enabled. Defaults to unset.

Unmanaged=

Takes a boolean. When "yes", no attempts are made to bring up or configure matching links, equivalent to when there are no matching network files. Defaults to "no".

This is useful for preventing later matching network files from interfering with certain interfaces that are fully controlled by other applications.

Group=

Link groups are similar to port ranges found in managed switches. When network interfaces are added to a numbered group, operations on all the interfaces from that group can be performed at once. Takes an unsigned integer in the range 0...4294967295. Defaults to unset.

RequiredForOnline=

Takes a boolean or a minimum operational state and an optional maximum operational state. Please see **networkctl**(1) for possible operational states. When "yes", the network is deemed required when determining whether the system is online (including when running **systemd–networkd–wait–online**). When "no", the network is ignored when determining the online state. When a minimum operational state and an optional maximum operational state are set, "yes" is implied, and this controls the minimum and maximum operational state required for the network interface to be considered online.

Defaults to "yes" when *ActivationPolicy*= is not set, or set to "up", "always-up", or "bound". Defaults to "no" when *ActivationPolicy*= is set to "manual" or "down". This is forced to "no" when *ActivationPolicy*= is set to "always-down".

The network will be brought up normally (as configured by *ActivationPolicy*=), but in the event that there is no address being assigned by DHCP or the cable is not plugged in, the link will simply remain offline and be skipped automatically by **systemd–networkd–wait–online** if "RequiredForOnline=no".

RequiredFamilyForOnline=

Takes an address family. When specified, an IP address in the given family is deemed required when determining whether the link is online (including when running **systemd–networkd–wait–online**). Takes one of "ipv4", "ipv6", "both", or "any". Defaults to "any". Note that this option has no effect if "RequiredForOnline=no", or if "RequiredForOnline=" specifies a minimum operational state below "degraded".

ActivationPolicy=

Specifies the policy for **systemd–networkd** managing the link administrative state. Specifically, this controls how **systemd–networkd** changes the network device's "IFF_UP" flag, which is sometimes controlled by system administrators by running e.g., **ip link set dev eth0 up** or **ip link set dev eth0 down**, and can also be changed with **networkctl up eth0** or **networkctl down eth0**.

Takes one of "up", "always-up", "manual", "always-down", "down", or "bound". When "manual", **systemd–networkd** will not change the link's admin state automatically; the system administrator must bring the interface up or down manually, as desired. When "up" (the default) or "always-up", or "down" or "always-down", **systemd–networkd** will set the link up or down, respectively, when the interface is (re)configured. When "always-up" or "always-down", **systemd–networkd** will set the link up or down, respectively, any time **systemd–networkd** detects a change in the administrative state. When *BindCarrier*= is also set, this is automatically set to "bound" and any other value is ignored.

When the policy is set to "down" or "manual", the default value of *RequiredForOnline*= is "no". When the policy is set to "always-down", the value of *RequiredForOnline*= forced to "no".

The administrative state is not the same as the carrier state, so using "always-up" does not mean the link will never lose carrier. The link carrier depends on both the administrative state as well as the network device's physical connection. However, to avoid reconfiguration failures, when using "always-up", *IgnoreCarrierLoss*= is forced to true.

[SR-IOV] SECTION OPTIONS

The [SR-IOV] section accepts the following keys. Specify several [SR-IOV] sections to configure several SR-IOVs. SR-IOV provides the ability to partition a single physical PCI resource into virtual PCI functions which can then be injected into a VM. In the case of network VFs, SR-IOV improves north-south network performance (that is, traffic with endpoints outside the host machine) by allowing traffic to bypass the host machine's network stack.

VirtualFunction=
 Specifies a Virtual Function (VF), lightweight PCIe function designed solely to move data in and out.
 Takes an unsigned integer in the range 0...2147483646. This option is compulsory.

VLANId=
 Specifies VLAN ID of the virtual function. Takes an unsigned integer in the range 1...4095.

QualityOfService=
 Specifies quality of service of the virtual function. Takes an unsigned integer in the range 1...4294967294.

VLANProtocol=
 Specifies VLAN protocol of the virtual function. Takes "802.1Q" or "802.1ad".

MACSpoofCheck=
 Takes a boolean. Controls the MAC spoof checking. When unset, the kernel's default will be used.

QueryReceiveSideScaling=
 Takes a boolean. Toggle the ability of querying the receive side scaling (RSS) configuration of the virtual function (VF). The VF RSS information like RSS hash key may be considered sensitive on some devices where this information is shared between VF and the physical function (PF). When unset, the kernel's default will be used.

Trust=
 Takes a boolean. Allows to set trust mode of the virtual function (VF). When set, VF users can set a specific feature which may impact security and/or performance. When unset, the kernel's default will be used.

LinkState=
 Allows to set the link state of the virtual function (VF). Takes a boolean or a special value "auto". Setting to "auto" means a reflection of the physical function (PF) link state, "yes" lets the VF to communicate with other VFs on this host even if the PF link state is down, "no" causes the hardware to drop any packets sent by the VF. When unset, the kernel's default will be used.

MACAddress=
 Specifies the MAC address for the virtual function.

[NETWORK] SECTION OPTIONS

The [Network] section accepts the following keys:

Description=
 A description of the device. This is only used for presentation purposes.

DHCP=
 Enables DHCPv4 and/or DHCPv6 client support. Accepts "yes", "no", "ipv4", or "ipv6". Defaults to "no".
 Note that DHCPv6 will by default be triggered by Router Advertisement, if that is enabled, regardless of this parameter. By enabling DHCPv6 support explicitly, the DHCPv6 client will be started regardless of the presence of routers on the link, or what flags the routers pass. See "IPv6AcceptRA=". Furthermore, note that by default the domain name specified through DHCP, on Ubuntu, are used for name resolution. See option **UseDomains**= below.
 See the [DHCPv4] or [DHCPv6] sections below for further configuration options for the DHCP client support.

DHCPServer=
 Takes a boolean. If set to "yes", DHCPv4 server will be started. Defaults to "no". Further settings for the DHCP server may be set in the [DHCPServer] section described below.

LinkLocalAddressing=

Enables link-local address autoconfiguration. Accepts **yes**, **no**, **ipv4**, and **ipv6**. An IPv6 link-local address is configured when **yes** or **ipv6**. An IPv4 link-local address is configured when **yes** or **ipv4** and when DHCPv4 autoconfiguration has been unsuccessful for some time. (IPv4 link-local address autoconfiguration will usually happen in parallel with repeated attempts to acquire a DHCPv4 lease).

Defaults to **no** when *Bridge*=**yes** is set, and **ipv6** otherwise.

IPv6LinkLocalAddressGenerationMode=

Specifies how IPv6 link local address is generated. Takes one of "eui64", "none", "stable-privacy" and "random". When unset, "stable-privacy" is used if *IPv6StableSecretAddress*= is specified, and if not, "eui64" is used. Note that if *LinkLocalAddressing*= is "no" or "ipv4", then

IPv6LinkLocalAddressGenerationMode= will be ignored. Also, even if *LinkLocalAddressing*= is "yes" or "ipv6", setting *IPv6LinkLocalAddressGenerationMode*=**none** disables to configure an IPv6 link-local address.

IPv6StableSecretAddress=

Takes an IPv6 address. The specified address will be used as a stable secret for generating IPv6 link-local address. If this setting is specified, and *IPv6LinkLocalAddressGenerationMode*= is unset, then *IPv6LinkLocalAddressGenerationMode*=**stable-privacy** is implied. If this setting is not specified, and "stable-privacy" is set to *IPv6LinkLocalAddressGenerationMode*=, then a stable secret address will be generated from the local machine ID and the interface name.

IPv4LLRoute=

Takes a boolean. If set to true, sets up the route needed for non-IPv4LL hosts to communicate with IPv4LL-only hosts. Defaults to false.

DefaultRouteOnDevice=

Takes a boolean. If set to true, sets up the default route bound to the interface. Defaults to false. This is useful when creating routes on point-to-point interfaces. This is equivalent to e.g. the following,

```
ip route add default dev veth99
```

or,

```
[Route]
Gateway=0.0.0.0
```

Currently, there are no way to specify e.g., the table for the route configured by this setting. To configure the default route with such an additional property, please use the following instead:

```
[Route]
Gateway=0.0.0.0
Table=1234
```

IPv6Token=

Specifies an optional address generation mode for the Stateless Address Autoconfiguration (SLAAC). Supported modes are "prefixstable" and "static".

When the mode is set to "static", an IPv6 address must be specified after a colon ":"), and the lower bits of the supplied address are combined with the upper bits of a prefix received in a Router Advertisement (RA) message to form a complete address. Note that if multiple prefixes are received in an RA message, or in multiple RA messages, addresses will be formed from each of them using the supplied address. This mode implements SLAAC but uses a static interface identifier instead of an identifier generated by using the EUI-64 algorithm. Because the interface identifier is static, if Duplicate Address Detection detects that the computed address is a duplicate (in use by another node on the link), then this mode will fail to provide an address for that prefix. If an IPv6 address without mode is specified, then "static" mode is assumed.

When the mode is set to "prefixstable" the [RFC 7217](#)^[1] algorithm for generating interface identifiers will be used. This mode can optionally take an IPv6 address separated with a colon (:). If an IPv6 address is specified, then an interface identifier is generated only when a prefix received in an RA message matches the supplied address.

If no address generation mode is specified (which is the default), or a received prefix does not match any of the addresses provided in "prefixstable" mode, then the EUI-64 algorithm will be used to form an interface identifier for that prefix. This mode is also SLAAC, but with a potentially stable interface identifier which does not directly map to the interface's hardware address.

Note that the "prefixstable" algorithm uses both the interface name and MAC address as input to the hash to compute the interface identifier, so if either of those are changed the resulting interface identifier (and address) will change, even if the prefix received in the RA message has not changed.

This setting can be specified multiple times. If an empty string is assigned, then all previous assignments are cleared.

Examples:

```
IPv6Token=::1a:2b:3c:4d
IPv6Token=static::1a:2b:3c:4d
IPv6Token=prefixstable
IPv6Token=prefixstable:2002:da8:1::
```

LLMNR=

Takes a boolean or "resolve". When true, enables [Link-Local Multicast Name Resolution](#)^[2] on the link. When set to "resolve", only resolution is enabled, but not host registration and announcement. Defaults to true. This setting is read by [systemd-resolved.service\(8\)](#).

MulticastDNS=

Takes a boolean or "resolve". When true, enables [Multicast DNS](#)^[3] support on the link. When set to "resolve", only resolution is enabled, but not host or service registration and announcement. Defaults to false. This setting is read by [systemd-resolved.service\(8\)](#).

DNSOverTLS=

Takes a boolean or "opportunistic". When true, enables [DNS-over-TLS](#)^[4] support on the link. When set to "opportunistic", compatibility with non-DNS-over-TLS servers is increased, by automatically turning off DNS-over-TLS servers in this case. This option defines a per-interface setting for [resolved.conf\(5\)](#)'s global *DNSOverTLS=* option. Defaults to false. This setting is read by [systemd-resolved.service\(8\)](#).

DNSSEC=

Takes a boolean or "allow-downgrade". When true, enables [DNSSEC](#)^[5] DNS validation support on the link. When set to "allow-downgrade", compatibility with non-DNSSEC capable networks is increased, by automatically turning off DNSSEC in this case. This option defines a per-interface setting for [resolved.conf\(5\)](#)'s global *DNSSEC=* option. Defaults to false. This setting is read by [systemd-resolved.service\(8\)](#).

DNSSECNegativeTrustAnchors=

A space-separated list of DNSSEC negative trust anchor domains. If specified and DNSSEC is enabled, look-ups done via the interface's DNS server will be subject to the list of negative trust anchors, and not require authentication for the specified domains, or anything below it. Use this to disable DNSSEC authentication for specific private domains, that cannot be proven valid using the Internet DNS hierarchy. Defaults to the empty list. This setting is read by [systemd-resolved.service\(8\)](#).

LLDP=

Controls support for Ethernet LLDP packet reception. LLDP is a link-layer protocol commonly

implemented on professional routers and bridges which announces which physical port a system is connected to, as well as other related data. Accepts a boolean or the special value "routers-only". When true, incoming LLDP packets are accepted and a database of all LLDP neighbors maintained. If "routers-only" is set only LLDP data of various types of routers is collected and LLDP data about other types of devices ignored (such as stations, telephones and others). If false, LLDP reception is disabled. Defaults to "routers-only". Use **networkctl**(1) to query the collected neighbor data. LLDP is only available on Ethernet links. See *EmitLLDP=* below for enabling LLDP packet emission from the local system.

EmitLLDP=

Controls support for Ethernet LLDP packet emission. Accepts a boolean parameter or the special values "nearest-bridge", "non-tpmr-bridge" and "customer-bridge". Defaults to false, which turns off LLDP packet emission. If not false, a short LLDP packet with information about the local system is sent out in regular intervals on the link. The LLDP packet will contain information about the local hostname, the local machine ID (as stored in **machine-id**(5)) and the local interface name, as well as the pretty hostname of the system (as set in **machine-info**(5)). LLDP emission is only available on Ethernet links. Note that this setting passes data suitable for identification of host to the network and should thus not be enabled on untrusted networks, where such identification data should not be made available. Use this option to permit other systems to identify on which interfaces they are connected to this system. The three special values control propagation of the LLDP packets. The "nearest-bridge" setting permits propagation only to the nearest connected bridge, "non-tpmr-bridge" permits propagation across Two-Port MAC Relays, but not any other bridges, and "customer-bridge" permits propagation until a customer bridge is reached. For details about these concepts, see [IEEE 802.1AB-2016^{\[6\]}](#). Note that configuring this setting to true is equivalent to "nearest-bridge", the recommended and most restricted level of propagation. See *LLDP=* above for an option to enable LLDP reception.

BindCarrier=

A link name or a list of link names. When set, controls the behavior of the current link. When all links in the list are in an operational down state, the current link is brought down. When at least one link has carrier, the current interface is brought up.

This forces *ActivationPolicy=* to be set to "bound".

Address=

A static IPv4 or IPv6 address and its prefix length, separated by a "/" character. Specify this key more than once to configure several addresses. The format of the address must be as described in **inet_pton**(3). This is a short-hand for an [Address] section only containing an Address key (see below). This option may be specified more than once.

If the specified address is "0.0.0.0" (for IPv4) or "::" (for IPv6), a new address range of the requested size is automatically allocated from a system-wide pool of unused ranges. Note that the prefix length must be equal or larger than 8 for IPv4, and 64 for IPv6. The allocated range is checked against all current network interfaces and all known network configuration files to avoid address range conflicts. The default system-wide pool consists of 192.168.0.0/16, 172.16.0.0/12 and 10.0.0.0/8 for IPv4, and fd00::/8 for IPv6. This functionality is useful to manage a large number of dynamically created network interfaces with the same network configuration and automatic address range assignment.

Gateway=

The gateway address, which must be in the format described in **inet_pton**(3). This is a short-hand for a [Route] section only containing a Gateway key. This option may be specified more than once.

DNS=

A DNS server address, which must be in the format described in **inet_pton**(3). This option may be specified more than once. Each address can optionally take a port number separated with ":"; a network interface name or index separated with "%", and a Server Name Indication (SNI) separated with "#". When IPv6 address is specified with a port number, then the address must be in the square

brackets. That is, the acceptable full formats are "111.222.333.444:9953%ifname#example.com" for IPv4 and "[1111:2222::3333]:9953%ifname#example.com" for IPv6. If an empty string is assigned, then all previous assignments are cleared. This setting is read by **systemd-resolved.service**(8).

Domains=

A whitespace-separated list of domains which should be resolved using the DNS servers on this link. Each item in the list should be a domain name, optionally prefixed with a tilde ("~"). The domains with the prefix are called "routing-only domains". The domains without the prefix are called "search domains" and are first used as search suffixes for extending single-label hostnames (hostnames containing no dots) to become fully qualified domain names (FQDNs). If a single-label hostname is resolved on this interface, each of the specified search domains are appended to it in turn, converting it into a fully qualified domain name, until one of them may be successfully resolved.

Both "search" and "routing-only" domains are used for routing of DNS queries: look-ups for hostnames ending in those domains (hence also single label names, if any "search domains" are listed), are routed to the DNS servers configured for this interface. The domain routing logic is particularly useful on multi-homed hosts with DNS servers serving particular private DNS zones on each interface.

The "routing-only" domain "~." (the tilde indicating definition of a routing domain, the dot referring to the DNS root domain which is the implied suffix of all valid DNS names) has special effect. It causes all DNS traffic which does not match another configured domain routing entry to be routed to DNS servers specified for this interface. This setting is useful to prefer a certain set of DNS servers if a link on which they are connected is available.

This setting is read by **systemd-resolved.service**(8). "Search domains" correspond to the *domain* and *search* entries in **resolv.conf**(5). Domain name routing has no equivalent in the traditional glibc API, which has no concept of domain name servers limited to a specific link.

DNSDefaultRoute=

Takes a boolean argument. If true, this link's configured DNS servers are used for resolving domain names that do not match any link's configured *Domains*= setting. If false, this link's configured DNS servers are never used for such domains, and are exclusively used for resolving names that match at least one of the domains configured on this link. If not specified defaults to an automatic mode: queries not matching any link's configured domains will be routed to this link if it has no routing-only domains configured.

NTP=

An NTP server address (either an IP address, or a hostname). This option may be specified more than once. This setting is read by **systemd-timesyncd.service**(8).

IPForward=

Configures IP packet forwarding for the system. If enabled, incoming packets on any network interface will be forwarded to any other interfaces according to the routing table. Takes a boolean, or the values "ipv4" or "ipv6", which only enable IP packet forwarding for the specified address family. This controls the net.ipv4.ip_forward and net.ipv6.conf.all.forwarding sysctl options of the network interface (see [ip-sysctl.txt](#)^[7] for details about sysctl options). Defaults to "no".

Note: this setting controls a global kernel option, and does so one way only: if a network that has this setting enabled is set up the global setting is turned on. However, it is never turned off again, even after all networks with this setting enabled are shut down again.

To allow IP packet forwarding only between specific network interfaces use a firewall.

IPMasquerade=

Configures IP masquerading for the network interface. If enabled, packets forwarded from the network interface will appear as coming from the local host. Takes one of "ipv4", "ipv6", "both", or "no".

Defaults to "no". If enabled, this automatically sets *IPForward=* to one of "ipv4", "ipv6" or "yes".

Note. Any positive boolean values such as "yes" or "true" are now deprecated. Please use one of the values in the above.

IPv6PrivacyExtensions=

Configures use of stateless temporary addresses that change over time (see [RFC 4941](#)^[8], Privacy Extensions for Stateless Address Autoconfiguration in IPv6). Takes a boolean or the special values "prefer-public" and "kernel". When true, enables the privacy extensions and prefers temporary addresses over public addresses. When "prefer-public", enables the privacy extensions, but prefers public addresses over temporary addresses. When false, the privacy extensions remain disabled. When "kernel", the kernel's default setting will be left in place. Defaults to "no".

IPv6AcceptRA=

Takes a boolean. Controls IPv6 Router Advertisement (RA) reception support for the interface. If true, RAs are accepted; if false, RAs are ignored. When RAs are accepted, they may trigger the start of the DHCPv6 client if the relevant flags are set in the RA data, or if no routers are found on the link. The default is to disable RA reception for bridge devices or when IP forwarding is enabled, and to enable it otherwise. Cannot be enabled on bond devices and when link local addressing is disabled.

Further settings for the IPv6 RA support may be configured in the [IPv6AcceptRA] section, see below.

Also see [ip-sysctl.txt](#)^[7] in the kernel documentation regarding "accept_ra", but note that systemd's setting of 1 (i.e. true) corresponds to kernel's setting of 2.

Note that kernel's implementation of the IPv6 RA protocol is always disabled, regardless of this setting. If this option is enabled, a userspace implementation of the IPv6 RA protocol is used, and the kernel's own implementation remains disabled, since **systemd-networkd** needs to know all details supplied in the advertisements, and these are not available from the kernel if the kernel's own implementation is used.

IPv6DuplicateAddressDetection=

Configures the amount of IPv6 Duplicate Address Detection (DAD) probes to send. When unset, the kernel's default will be used.

IPv6HopLimit=

Configures IPv6 Hop Limit. For each router that forwards the packet, the hop limit is decremented by 1. When the hop limit field reaches zero, the packet is discarded. When unset, the kernel's default will be used.

IPv4AcceptLocal=

Takes a boolean. Accept packets with local source addresses. In combination with suitable routing, this can be used to direct packets between two local interfaces over the wire and have them accepted properly. When unset, the kernel's default will be used.

IPv4RouteLocalnet=

Takes a boolean. When true, the kernel does not consider loopback addresses as martian source or destination while routing. This enables the use of 127.0.0.0/8 for local routing purposes. When unset, the kernel's default will be used.

IPv4ProxyARP=

Takes a boolean. Configures proxy ARP for IPv4. Proxy ARP is the technique in which one host, usually a router, answers ARP requests intended for another machine. By "faking" its identity, the router accepts responsibility for routing packets to the "real" destination. See [RFC 1027](#)^[9]. When unset, the kernel's default will be used.

IPv6ProxyNDP=

Takes a boolean. Configures proxy NDP for IPv6. Proxy NDP (Neighbor Discovery Protocol) is a technique for IPv6 to allow routing of addresses to a different destination when peers expect them to

be present on a certain physical link. In this case a router answers Neighbour Advertisement messages intended for another machine by offering its own MAC address as destination. Unlike proxy ARP for IPv4, it is not enabled globally, but will only send Neighbour Advertisement messages for addresses in the IPv6 neighbor proxy table, which can also be shown by **ip -6 neighbour show proxy**.
 systemd–networkd will control the per-interface ‘proxy_ndp’ switch for each configured interface depending on this option. When unset, the kernel’s default will be used.

IPv6ProxyNDPAddress=

An IPv6 address, for which Neighbour Advertisement messages will be proxied. This option may be specified more than once. systemd–networkd will add the **IPv6ProxyNDPAddress=** entries to the kernel’s IPv6 neighbor proxy table. This option implies **IPv6ProxyNDP=yes** but has no effect if **IPv6ProxyNDP** has been set to false. When unset, the kernel’s default will be used.

IPv6SendRA=

Whether to enable or disable Router Advertisement sending on a link. Takes a boolean value. When enabled, prefixes configured in [IPv6Prefix] sections and routes configured in [IPv6RoutePrefix] sections are distributed as defined in the [IPv6SendRA] section. If **DHCPv6PrefixDelegation=** is enabled, then the delegated prefixes are also distributed. See **DHCPv6PrefixDelegation=** setting and the [IPv6SendRA], [IPv6Prefix], [IPv6RoutePrefix], and [DHCPv6PrefixDelegation] sections for more configuration options.

DHCPv6PrefixDelegation=

Takes a boolean value. When enabled, requests prefixes using a DHCPv6 client configured on another link. By default, an address within each delegated prefix will be assigned, and the prefixes will be announced through IPv6 Router Advertisement when *IPv6SendRA=* is enabled. Such default settings can be configured in [DHCPv6PrefixDelegation] section. Defaults to disabled.

IPv6MTUBytes=

Configures IPv6 maximum transmission unit (MTU). An integer greater than or equal to 1280 bytes. When unset, the kernel’s default will be used.

BatmanAdvanced=, Bond=, Bridge=, VRF=

The name of the B.A.T.M.A.N. Advanced, bond, bridge, or VRF interface to add the link to. See **systemd.netdev(5)**.

IPVLAN=, IPVLTAP=, L2TP=, MACsec=, MACVLAN=, MACVTAP=, Tunnel=, VLAN=, VXLAN=, Xfrm=

The name of an IPVLAN, IPVLTAP, L2TP, MACsec, MACVLAN, MACVTAP, tunnel, VLAN, VXLAN, or Xfrm to be created on the link. See **systemd.netdev(5)**. This option may be specified more than once.

ActiveSlave=

Takes a boolean. Specifies the new active slave. The "ActiveSlave=" option is only valid for following modes: "active-backup", "balance-alb" and "balance-tlb". Defaults to false.

PrimarySlave=

Takes a boolean. Specifies which slave is the primary device. The specified device will always be the active slave while it is available. Only when the primary is off-line will alternate devices be used. This is useful when one slave is preferred over another, e.g. when one slave has higher throughput than another. The "PrimarySlave=" option is only valid for following modes: "active-backup", "balance-alb" and "balance-tlb". Defaults to false.

ConfigureWithoutCarrier=

Takes a boolean. Allows networkd to configure a specific link even if it has no carrier. Defaults to false. If **IgnoreCarrierLoss=** is not explicitly set, it will default to this value.

IgnoreCarrierLoss=

Takes a boolean. Allows networkd to retain both the static and dynamic configuration of the interface even if its carrier is lost. When unset, the value specified with **ConfigureWithoutCarrier=** is used.

When **ActivationPolicy=** is set to "always-up", this is forced to "true".

KeepConfiguration=

Takes a boolean or one of "static", "dhcp-on-stop", "dhcp". When "static", **systemd-networkd** will not drop static addresses and routes on starting up process. When set to "dhcp-on-stop", **systemd-networkd** will not drop addresses and routes on stopping the daemon. When "dhcp", the addresses and routes provided by a DHCP server will never be dropped even if the DHCP lease expires. This is contrary to the DHCP specification, but may be the best choice if, e.g., the root filesystem relies on this connection. The setting "dhcp" implies "dhcp-on-stop", and "yes" implies "dhcp" and "static". Defaults to "no".

[ADDRESS] SECTION OPTIONS

An [Address] section accepts the following keys. Specify several [Address] sections to configure several addresses.

Address=

As in the [Network] section. This key is mandatory. Each [Address] section can contain one *Address=* setting.

Peer=

The peer address in a point-to-point connection. Accepts the same format as the *Address=* key.

Broadcast=

Takes an IPv4 address or boolean value. The address must be in the format described in **inet_nton(3)**. If set to true, then the IPv4 broadcast address will be derived from the *Address=* setting. If set to false, then the broadcast address will not be set. Defaults to true, except for wireguard interfaces, where it default to false.

Label=

An address label.

PreferredLifetime=

Allows the default "preferred lifetime" of the address to be overridden. Only three settings are accepted: "forever", "infinity", which is the default and means that the address never expires, and "0", which means that the address is considered immediately "expired" and will not be used, unless explicitly requested. A setting of **PreferredLifetime=0** is useful for addresses which are added to be used only by a specific application, which is then configured to use them explicitly.

Scope=

The scope of the address, which can be "global" (valid everywhere on the network, even through a gateway), "link" (only valid on this device, will not traverse a gateway) or "host" (only valid within the device itself, e.g. 127.0.0.1) or an unsigned integer in the range 0...255. Defaults to "global".

RouteMetric=

The metric of the prefix route, which is pointing to the subnet of the configured IP address, taking the configured prefix length into account. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used. This setting will be ignored when *AddPrefixRoute=* is false.

HomeAddress=

Takes a boolean. Designates this address the "home address" as defined in [RFC 6275](#)^[10]. Supported only on IPv6. Defaults to false.

DuplicateAddressDetection=

Takes one of "ipv4", "ipv6", "both", "none". When "ipv4", performs IPv4 Address Conflict Detection. See [RFC 5227](#)^[11]. When "ipv6", performs IPv6 Duplicate Address Detection. See [RFC 4862](#)^[12]. Defaults to "ipv4" for IPv4 link-local addresses, "ipv6" for IPv6 addresses, and "none" otherwise.

ManageTemporaryAddress=

Takes a boolean. If true the kernel manage temporary addresses created from this one as template on behalf of Privacy Extensions [RFC 3041](#)^[13]. For this to become active, the `use_tempaddr` sysctl setting has to be set to a value greater than zero. The given address needs to have a prefix length of 64. This flag allows using privacy extensions in a manually configured network, just like if stateless

auto-configuration was active. Defaults to false.

AddPrefixRoute=

Takes a boolean. When true, the prefix route for the address is automatically added. Defaults to true.

AutoJoin=

Takes a boolean. Joining multicast group on ethernet level via **ip maddr** command would not work if we have an Ethernet switch that does IGMP snooping since the switch would not replicate multicast packets on ports that did not have IGMP reports for the multicast addresses. Linux vxlan interfaces created via **ip link add vxlan** or networkd's netdev kind vxlan have the group option that enables them to do the required join. By extending ip address command with option "autojoin" we can get similar functionality for openvswitch (OVS) vxlan interfaces as well as other tunneling mechanisms that need to receive multicast traffic. Defaults to "no".

[NEIGHBOR] SECTION OPTIONS

A [Neighbor] section accepts the following keys. The neighbor section adds a permanent, static entry to the neighbor table (IPv6) or ARP table (IPv4) for the given hardware address on the links matched for the network. Specify several [Neighbor] sections to configure several static neighbors.

Address=

The IP address of the neighbor.

LinkLayerAddress=

The link layer address (MAC address or IP address) of the neighbor.

[IPV6ADDRESSLABEL] SECTION OPTIONS

An [IPv6AddressLabel] section accepts the following keys. Specify several [IPv6AddressLabel] sections to configure several address labels. IPv6 address labels are used for address selection. See [RFC 3484^{\[14\]}](#). Precedence is managed by userspace, and only the label itself is stored in the kernel.

Label=

The label for the prefix, an unsigned integer in the range 0–4294967294. 0xffffffff is reserved. This setting is mandatory.

Prefix=

IPv6 prefix is an address with a prefix length, separated by a slash "/" character. This key is mandatory.

[ROUTINGPOLICYRULE] SECTION OPTIONS

An [RoutingPolicyRule] section accepts the following keys. Specify several [RoutingPolicyRule] sections to configure several rules.

TypeOfService=

Takes a number between 0 and 255 that specifies the type of service to match.

From=

Specifies the source address prefix to match. Possibly followed by a slash and the prefix length.

To=

Specifies the destination address prefix to match. Possibly followed by a slash and the prefix length.

FirewallMark=

Specifies the iptables firewall mark value to match (a number between 1 and 4294967295). Optionally, the firewall mask (also a number between 1 and 4294967295) can be suffixed with a slash ("/"), e.g., "7/255".

Table=

Specifies the routing table identifier to lookup if the rule selector matches. Takes one of predefined names "default", "main", and "local", and names defined in *RouteTable*= in **networkd.conf(5)**, or a number between 1 and 4294967295. Defaults to "main".

Priority=

Specifies the priority of this rule. *Priority*= is an unsigned integer in the range 0...4294967295.

Higher number means lower priority, and rules get processed in order of increasing number. Defaults to unset, and the kernel will pick a value dynamically.

IncomingInterface=

Specifies incoming device to match. If the interface is loopback, the rule only matches packets originating from this host.

OutgoingInterface=

Specifies the outgoing device to match. The outgoing interface is only available for packets originating from local sockets that are bound to a device.

SourcePort=

Specifies the source IP port or IP port range match in forwarding information base (FIB) rules. A port range is specified by the lower and upper port separated by a dash. Defaults to unset.

DestinationPort=

Specifies the destination IP port or IP port range match in forwarding information base (FIB) rules. A port range is specified by the lower and upper port separated by a dash. Defaults to unset.

IPProtocol=

Specifies the IP protocol to match in forwarding information base (FIB) rules. Takes IP protocol name such as "tcp", "udp" or "sctp", or IP protocol number such as "6" for "tcp" or "17" for "udp". Defaults to unset.

InvertRule=

A boolean. Specifies whether the rule is to be inverted. Defaults to false.

Family=

Takes a special value "ipv4", "ipv6", or "both". By default, the address family is determined by the address specified in *To*= or *From*=. If neither *To*= nor *From*= are specified, then defaults to "ipv4".

User=

Takes a username, a user ID, or a range of user IDs separated by a dash. Defaults to unset.

SuppressPrefixLength=

Takes a number *N* in the range 0...128 and rejects routing decisions that have a prefix length of *N* or less. Defaults to unset.

Type=

Specifies Routing Policy Database (RPDB) rule type. Takes one of "blackhole", "unreachable" or "prohibit".

[NEXTHOP] SECTION OPTIONS

The [NextHop] section is used to manipulate entries in the kernel's "nexthop" tables. The [NextHop] section accepts the following keys. Specify several [NextHop] sections to configure several hops.

Id=

The id of the next hop. Takes an unsigned integer in the range 1...4294967295. If left unspecified, then automatically chosen by kernel.

Gateway=

As in the [Network] section.

Family=

Takes one of the special values "ipv4" or "ipv6". By default, the family is determined by the address specified in *Gateway*=. If *Gateway*= is not specified, then defaults to "ipv4".

OnLink=

Takes a boolean. If set to true, the kernel does not have to check if the gateway is reachable directly by the current machine (i.e., attached to the local network), so that we can insert the nexthop in the kernel table without it being complained about. Defaults to "no".

Blackhole=

Takes a boolean. If enabled, packets to the corresponding routes are discarded silently, and *Gateway*=

cannot be specified. Defaults to "no".

Group=

Takes a whitespace separated list of nexthop IDs. Each ID must be in the range 1...4294967295.

Optionally, each nexthop ID can take a weight after a colon ("id[:weight]"). The weight must be in the range 1...255. If the weight is not specified, then it is assumed that the weight is 1. This setting cannot be specified with *Gateway=*, *Family=*, *Blackhole=*. This setting can be specified multiple times. If an empty string is assigned, then all previous assignments are cleared. Defaults to unset.

[ROUTE] SECTION OPTIONS

The [Route] section accepts the following keys. Specify several [Route] sections to configure several routes.

Gateway=

Takes the gateway address or the special values "_dhcp4" and "_ipv6ra". If "_dhcp4" or "_ipv6ra" is set, then the gateway address provided by DHCPv4 or IPv6 RA is used.

GatewayOnLink=

Takes a boolean. If set to true, the kernel does not have to check if the gateway is reachable directly by the current machine (i.e., attached to the local network), so that we can insert the route in the kernel table without it being complained about. Defaults to "no".

Destination=

The destination prefix of the route. Possibly followed by a slash and the prefix length. If omitted, a full-length host route is assumed.

Source=

The source prefix of the route. Possibly followed by a slash and the prefix length. If omitted, a full-length host route is assumed.

Metric=

The metric of the route. Takes an unsigned integer in the range 0...4294967295. Defaults to unset, and the kernel's default will be used.

IPv6Preference=

Specifies the route preference as defined in [RFC 4191](#)^[15] for Router Discovery messages. Which can be one of "low" the route has a lowest priority, "medium" the route has a default priority or "high" the route has a highest priority.

Scope=

The scope of the IPv4 route, which can be "global", "site", "link", "host", or "nowhere":

- "global" means the route can reach hosts more than one hop away.
- "site" means an interior route in the local autonomous system.
- "link" means the route can only reach hosts on the local network (one hop away).
- "host" means the route will not leave the local machine (used for internal addresses like 127.0.0.1).
- "nowhere" means the destination doesn't exist.

For IPv4 route, defaults to "host" if *Type=* is "local" or "nat", and "link" if *Type=* is "broadcast", "multicast", or "anycast". In other cases, defaults to "global". The value is not used for IPv6.

PreferredSource=

The preferred source address of the route. The address must be in the format described in [inet_nton\(3\)](#).

Table=

The table identifier for the route. Takes one of predefined names "default", "main", and "local", and names defined in *RouteTable=* in [networkd.conf\(5\)](#), or a number between 1 and 4294967295. The table can be retrieved using **ip route show table num**. If unset and *Type=* is "local", "broadcast", "anycast", or "nat", then "local" is used. In other cases, defaults to "main".

Protocol=

The protocol identifier for the route. Takes a number between 0 and 255 or the special values "kernel", "boot", "static", "ra" and "dhcp". Defaults to "static".

Type=

Specifies the type for the route. Takes one of "unicast", "local", "broadcast", "anycast", "multicast", "blackhole", "unreachable", "prohibit", "throw", "nat", and "xresolve". If "unicast", a regular route is defined, i.e. a route indicating the path to take to a destination network address. If "blackhole", packets to the defined route are discarded silently. If "unreachable", packets to the defined route are discarded and the ICMP message "Host Unreachable" is generated. If "prohibit", packets to the defined route are discarded and the ICMP message "Communication Administratively Prohibited" is generated. If "throw", route lookup in the current routing table will fail and the route selection process will return to Routing Policy Database (RPDB). Defaults to "unicast".

InitialCongestionWindow=

The TCP initial congestion window is used during the start of a TCP connection. During the start of a TCP session, when a client requests a resource, the server's initial congestion window determines how many packets will be sent during the initial burst of data without waiting for acknowledgement. Takes a number between 1 and 1023. Note that 100 is considered an extremely large value for this option. When unset, the kernel's default (typically 10) will be used.

InitialAdvertisedReceiveWindow=

The TCP initial advertised receive window is the amount of receive data (in bytes) that can initially be buffered at one time on a connection. The sending host can send only that amount of data before waiting for an acknowledgment and window update from the receiving host. Takes a number between 1 and 1023. Note that 100 is considered an extremely large value for this option. When unset, the kernel's default will be used.

QuickAck=

Takes a boolean. When true enables TCP quick ack mode for the route. When unset, the kernel's default will be used.

FastOpenNoCookie=

Takes a boolean. When true enables TCP fastopen without a cookie on a per-route basis. When unset, the kernel's default will be used.

TTLPropagate=

Takes a boolean. When true enables TTL propagation at Label Switched Path (LSP) egress. When unset, the kernel's default will be used.

MTUBytes=

The maximum transmission unit in bytes to set for the route. The usual suffixes K, M, G, are supported and are understood to the base of 1024.

Note that if IPv6 is enabled on the interface, and the MTU is chosen below 1280 (the minimum MTU for IPv6) it will automatically be increased to this value.

IPServiceType=

Takes string; "CS6" or "CS4". Used to set IP service type to CS6 (network control) or CS4 (Realtime). Defaults to CS6.

TCPAdvertisedMaximumSegmentSize=

Specifies the Path MSS (in bytes) hints given on TCP layer. The usual suffixes K, M, G, are supported and are understood to the base of 1024. An unsigned integer in the range 1–4294967294. When unset, the kernel's default will be used.

MultiPathRoute=address[@name] [weight]

Configures multipath route. Multipath routing is the technique of using multiple alternative paths through a network. Takes gateway address. Optionally, takes a network interface name or index separated with "@", and a weight in 1..256 for this multipath route separated with whitespace. This

setting can be specified multiple times. If an empty string is assigned, then all previous assignments are cleared.

NextHop=

Specifies the nexthop id. Takes an unsigned integer in the range 1...4294967295. If set, the corresponding [NextHop] section must be configured. Defaults to unset.

[DHCPV4] SECTION OPTIONS

The [DHCPv4] section configures the DHCPv4 client, if it is enabled with the *DHCP=* setting described above:

SendHostname=

When true (the default), the machine's hostname (or the value specified with *Hostname=*, described below) will be sent to the DHCP server. Note that the hostname must consist only of 7-bit ASCII lower-case characters and no spaces or dots, and be formatted as a valid DNS domain name. Otherwise, the hostname is not sent even if this option is true.

Hostname=

Use this value for the hostname which is sent to the DHCP server, instead of machine's hostname. Note that the specified hostname must consist only of 7-bit ASCII lower-case characters and no spaces or dots, and be formatted as a valid DNS domain name.

MUDURL=

When configured, the specified Manufacturer Usage Description (MUD) URL will be sent to the DHCPv4 server. Takes a URL of length up to 255 characters. A superficial verification that the string is a valid URL will be performed. DHCPv4 clients are intended to have at most one MUD URL associated with them. See [RFC 8520](#)^[16].

MUD is an embedded software standard defined by the IETF that allows IoT device makers to advertise device specifications, including the intended communication patterns for their device when it connects to the network. The network can then use this to author a context-specific access policy, so the device functions only within those parameters.

ClientIdentifier=

The DHCPv4 client identifier to use. Takes one of **mac**, **duid** or **duid-only**. If set to **mac**, the MAC address of the link is used. If set to **duid**, an RFC4361-compliant Client ID, which is the combination of IAID and DUID (see below), is used. If set to **duid-only**, only DUID is used, this may not be RFC compliant, but some setups may require to use this. Defaults to **duid**.

VendorClassIdentifier=

The vendor class identifier used to identify vendor type and configuration.

UserClass=

A DHCPv4 client can use UserClass option to identify the type or category of user or applications it represents. The information contained in this option is a string that represents the user class of which the client is a member. Each class sets an identifying string of information to be used by the DHCP service to classify clients. Takes a whitespace-separated list of strings.

DUIDType=

Override the global *DUIDType=* setting for this network. See **networkd.conf(5)** for a description of possible values.

DUIDRawData=

Override the global *DUIDRawData=* setting for this network. See **networkd.conf(5)** for a description of possible values.

IAID=

The DHCP Identity Association Identifier (IAID) for the interface, a 32-bit unsigned integer.

Anonymize=

Takes a boolean. When true, the options sent to the DHCP server will follow the [RFC 7844](#)^[17] (Anonymity Profiles for DHCP Clients) to minimize disclosure of identifying information. Defaults to

false.

This option should only be set to true when *MACAddressPolicy*= is set to **random** (see [systemd.link\(5\)](#)).

When true, *SendHostname*=, *ClientIdentifier*=, *VendorClassIdentifier*=, *UserClass*=, *RequestOptions*=, *SendOption*=, *SendVendorOption*=, and *MUDURL*= are ignored.

With this option enabled DHCP requests will mimic those generated by Microsoft Windows, in order to reduce the ability to fingerprint and recognize installations. This means DHCP request sizes will grow and lease data will be more comprehensive than normally, though most of the requested data is not actually used.

RequestOptions=

Sets request options to be sent to the server in the DHCPv4 request options list. A whitespace-separated list of integers in the range 1...254. Defaults to unset.

SendOption=

Send an arbitrary raw option in the DHCPv4 request. Takes a DHCP option number, data type and data separated with a colon ("option:type:value"). The option number must be an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using [C-style escapes](#)^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

SendVendorOption=

Send an arbitrary vendor option in the DHCPv4 request. Takes a DHCP option number, data type and data separated with a colon ("option:type:value"). The option number must be an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using [C-style escapes](#)^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

UseDNS=

When true (the default), the DNS servers received from the DHCP server will be used.

This corresponds to the **nameserver** option in [resolv.conf\(5\)](#).

RoutesToDNS=

When true, the routes to the DNS servers received from the DHCP server will be configured. When *UseDNS*= is disabled, this setting is ignored. Defaults to true.

UseNTP=

When true (the default), the NTP servers received from the DHCP server will be used by [systemd-timesyncd.service](#).

RoutesToNTP=

When true, the routes to the NTP servers received from the DHCP server will be configured. When *UseNTP*= is disabled, this setting is ignored. Defaults to true.

UseSIP=

When true (the default), the SIP servers received from the DHCP server will be collected and made available to client programs.

UseMTU=

When true, the interface maximum transmission unit from the DHCP server will be used on the current link. If *MTUBytes*= is set, then this setting is ignored. Defaults to false.

UseHostname=

When true (the default), the hostname received from the DHCP server will be set as the transient

hostname of the system.

UseDomains=

Takes a boolean, or the special value **route**. When true, the domain name received from the DHCP server will be used as DNS search domain over this link, similar to the effect of the **Domains=** setting. If set to **route**, the domain name received from the DHCP server will be used for routing DNS queries only, but not for searching, similar to the effect of the **Domains=** setting when the argument is prefixed with `"~"`. Defaults to true on Ubuntu.

It is recommended to enable this option only on trusted networks, as setting this affects resolution of all hostnames, in particular of single-label names. It is generally safer to use the supplied domain only as routing domain, rather than as search domain, in order to not have it affect local resolution of single-label names.

When set to true, this setting corresponds to the **domain** option in **resolv.conf(5)**.

UseRoutes=

When true (the default), the static routes will be requested from the DHCP server and added to the routing table with a metric of 1024, and a scope of **global**, **link** or **host**, depending on the route's destination and gateway. If the destination is on the local host, e.g., 127.x.x.x, or the same as the link's own address, the scope will be set to **host**. Otherwise if the gateway is null (a direct route), a **link** scope will be used. For anything else, scope defaults to **global**.

RouteMetric=

Set the routing metric for routes specified by the DHCP server. Takes an unsigned integer in the range 0...4294967295. Defaults to 1024.

RouteTable=num

The table identifier for DHCP routes (a number between 1 and 4294967295, or 0 to unset). The table can be retrieved using **ip route show table num**.

When used in combination with **VRF=**, the VRF's routing table is used when this parameter is not specified.

RouteMTUBytes=

Specifies the MTU for the DHCP routes. Please see the [Route] section for further details.

UseGateway=

When true, the gateway will be requested from the DHCP server and added to the routing table with a metric of 1024, and a scope of **link**. When unset, the value specified with *UseRoutes=* is used.

UseTimezone=

When true, the timezone received from the DHCP server will be set as timezone of the local system. Defaults to false.

FallbackLeaseLifetimeSec=

Allows to set DHCPv4 lease lifetime when DHCPv4 server does not send the lease lifetime. Takes one of "forever" or "infinity". The latter means that the address never expires. Defaults to unset.

RequestBroadcast=

Request the server to use broadcast messages before the IP address has been configured. This is necessary for devices that cannot receive RAW packets, or that cannot receive packets at all before an IP address has been configured. On the other hand, this must not be enabled on networks where broadcasts are filtered out.

MaxAttempts=

Specifies how many times the DHCPv4 client configuration should be attempted. Takes a number or "infinity". Defaults to "infinity". Note that the time between retries is increased exponentially, up to approximately one per minute, so the network will not be overloaded even if this number is high. The default is suitable in most circumstances.

ListenPort=

Set the port from which the DHCP client packets originate.

DenyList=

A whitespace-separated list of IPv4 addresses. DHCP offers from servers in the list are rejected. Note that if *AllowList=* is configured then *DenyList=* is ignored.

AllowList=

A whitespace-separated list of IPv4 addresses. DHCP offers from servers in the list are accepted.

SendRelease=

When true, the DHCPv4 client sends a DHCP release packet when it stops. Defaults to true.

SendDecline=

A boolean. When "true", the DHCPv4 client receives the IP address from the DHCP server. After a new IP is received, the DHCPv4 client performs IPv4 Duplicate Address Detection. If duplicate use is detected, the DHCPv4 client rejects the IP by sending a **DHCPDECLINE** packet and tries to obtain an IP address again. See [RFC 5224](#)^[11]. Defaults to "unset".

[DHCPV6] SECTION OPTIONS

The [DHCPv6] section configures the DHCPv6 client, if it is enabled with the *DHCP=* setting described above, or invoked by the IPv6 Router Advertisement:

MUDURL=, IAID=, DUIDType=, DUIDRawData=, RequestOptions=

As in the [DHCPv4] section.

SendOption=

As in the [DHCPv4] section, however because DHCPv6 uses 16-bit fields to store option numbers, the option number is an integer in the range 1...65536.

SendVendorOption=

Send an arbitrary vendor option in the DHCPv6 request. Takes an enterprise identifier, DHCP option number, data type, and data separated with a colon ("enterprise identifier:option:type:value"). Enterprise identifier is an unsigned integer in the range 1...4294967294. The option number must be an integer in the range 1...254. Data type takes one of "uint8", "uint16", "uint32", "ipv4address", "ipv6address", or "string". Special characters in the data string may be escaped using **C-style escapes**^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

UserClass=

A DHCPv6 client can use User Class option to identify the type or category of user or applications it represents. The information contained in this option is a string that represents the user class of which the client is a member. Each class sets an identifying string of information to be used by the DHCP service to classify clients. Special characters in the data string may be escaped using **C-style escapes**^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Takes a whitespace-separated list of strings. Note that currently **NUL** bytes are not allowed.

VendorClass=

A DHCPv6 client can use VendorClass option to identify the vendor that manufactured the hardware on which the client is running. The information contained in the data area of this option is contained in one or more opaque fields that identify details of the hardware configuration. Takes a whitespace-separated list of strings.

PrefixDelegationHint=

Takes an IPv6 address with prefix length in the same format as the *Address=* in the [Network] section. The DHCPv6 client will include a prefix hint in the DHCPv6 solicitation sent to the server. The prefix length must be in the range 1–128. Defaults to unset.

UseAddress=

When true (the default), the IP addresses provided by the DHCPv6 server will be assigned.

UseDNS=, *UseNTP=*, *UseHostname=*, *UseDomains=*
As in the [DHCPv4] section.

ForceDHCPv6PDOtherInformation=

Takes a boolean that enforces DHCPv6 stateful mode when the 'Other information' bit is set in Router Advertisement messages. By default setting only the 'O' bit in Router Advertisements makes DHCPv6 request network information in a stateless manner using a two-message Information Request and Information Reply message exchange. [RFC 7084](#)^[19], requirement WPD-4, updates this behavior for a Customer Edge router so that stateful DHCPv6 Prefix Delegation is also requested when only the 'O' bit is set in Router Advertisements. This option enables such a CE behavior as it is impossible to automatically distinguish the intention of the 'O' bit otherwise. By default this option is set to false, enable it if no prefixes are delegated when the device should be acting as a CE router.

WithoutRA=

Allows DHCPv6 client to start without router advertisements's managed or other address configuration flag. Takes one of "solicit" or "information-request". Defaults to unset.

RapidCommit=

Takes a boolean. The DHCPv6 client can obtain configuration parameters from a DHCPv6 server through a rapid two-message exchange (solicit and reply). When the rapid commit option is enabled by both the DHCPv6 client and the DHCPv6 server, the two-message exchange is used, rather than the default four-message exchange (solicit, advertise, request, and reply). The two-message exchange provides faster client configuration and is beneficial in environments in which networks are under a heavy load. See [RFC 3315](#)^[20] for details. Defaults to true.

[DHCPV6PREFIXDELEGATION] SECTION OPTIONS

The [DHCPv6PrefixDelegation] section configures delegated prefixes assigned by DHCPv6 server. The settings in this section are used only when *DHCPv6PrefixDelegation=* setting is enabled.

SubnetId=

Configure a specific subnet ID on the interface from a (previously) received prefix delegation. You can either set "auto" (the default) or a specific subnet ID (as defined in [RFC 4291](#)^[21], section 2.5.4), in which case the allowed value is hexadecimal, from 0 to 0xfffffffffffff inclusive.

Announce=

Takes a boolean. When enabled, and *IPv6SendRA=* in [Network] section is enabled, the delegated prefixes are distributed through the IPv6 Router Advertisement. Defaults to yes.

Assign=

Takes a boolean. Specifies whether to add an address from the delegated prefixes which are received from the WAN interface by the DHCPv6 Prefix Delegation. When true (on LAN interface), the EUI-64 algorithm will be used by default to form an interface identifier from the delegated prefixes. See also *Token=* setting below. Defaults to yes.

Token=

Specifies an optional address generation mode for assigning an address in each delegated prefix. Takes an IPv6 address. When set, the lower bits of the supplied address is combined with the upper bits of each delegatad prefix received from the WAN interface by the DHCPv6 Prefix Delegation to form a complete address. When *Assign=* is disabled, this setting is ignored. When unset, the EUI-64 algorithm will be used to form addresses. Defaults to unset.

ManageTemporaryAddress=

As in the [Address] section, but defaults to true.

RouteMetric=

The metric of the route to the delegated prefix subnet. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used.

[IPV6ACCEPTRA] SECTION OPTIONS

The [IPv6AcceptRA] section configures the IPv6 Router Advertisement (RA) client, if it is enabled with the *IPv6AcceptRA=* setting described above:

UseDNS=

When true (the default), the DNS servers received in the Router Advertisement will be used.

This corresponds to the **nameserver** option in **resolv.conf(5)**.

UseDomains=

Takes a boolean, or the special value "route". When true, the domain name received via IPv6 Router Advertisement (RA) will be used as DNS search domain over this link, similar to the effect of the **Domains=** setting. If set to "route", the domain name received via IPv6 RA will be used for routing DNS queries only, but not for searching, similar to the effect of the **Domains=** setting when the argument is prefixed with "~". Defaults to true on Ubuntu.

It is recommended to enable this option only on trusted networks, as setting this affects resolution of all hostnames, in particular of single-label names. It is generally safer to use the supplied domain only as routing domain, rather than as search domain, in order to not have it affect local resolution of single-label names.

When set to true, this setting corresponds to the **domain** option in **resolv.conf(5)**.

RouteTable=num

The table identifier for the routes received in the Router Advertisement (a number between 1 and 4294967295, or 0 to unset). The table can be retrieved using **ip route show table num**.

RouteMetric=

Set the routing metric for the routes received in the Router Advertisement. Takes an unsigned integer in the range 0...4294967295. Defaults to 1024.

UseAutonomousPrefix=

When true (the default), the autonomous prefix received in the Router Advertisement will be used and take precedence over any statically configured ones.

UseOnLinkPrefix=

When true (the default), the onlink prefix received in the Router Advertisement will be used and takes precedence over any statically configured ones.

RouterDenyList=

A whitespace-separated list of IPv6 router addresses. Any information advertised by the listed router is ignored.

RouterAllowList=

A whitespace-separated list of IPv6 router addresses. Only information advertised by the listed router is accepted. Note that if *RouterAllowList=* is configured then *RouterDenyList=* is ignored.

PrefixDenyList=

A whitespace-separated list of IPv6 prefixes. IPv6 prefixes supplied via router advertisements in the list are ignored.

PrefixAllowList=

A whitespace-separated list of IPv6 prefixes. IPv6 prefixes supplied via router advertisements in the list are allowed. Note that if *PrefixAllowList=* is configured then *PrefixDenyList=* is ignored.

RouteDenyList=

A whitespace-separated list of IPv6 route prefixes. IPv6 route prefixes supplied via router advertisements in the list are ignored.

RouteAllowList=

A whitespace-separated list of IPv6 route prefixes. IPv6 route prefixes supplied via router advertisements in the list are allowed. Note that if *RouteAllowList=* is configured then *RouteDenyList=* is ignored.

DHCPv6Client=

Takes a boolean, or the special value "always". When true or "always", the DHCPv6 client will be

started when the RA has the managed or other information flag. If set to "always", the DHCPv6 client will also be started in managed mode when neither managed nor other information flag is set in the RA. Defaults to true.

[DHCPSERVER] SECTION OPTIONS

The [DHCPServer] section contains settings for the DHCP server, if enabled via the *DHCPServer=* option described above:

ServerAddress=

Specifies server address for the DHCP server. Takes an IPv4 address with prefix length, for example "192.168.0.1/24". This setting may be useful when the link on which the DHCP server is running has multiple static addresses. When unset, one of static addresses in the link will be automatically selected. Defaults to unset.

PoolOffset=, PoolSize=

Configures the pool of addresses to hand out. The pool is a contiguous sequence of IP addresses in the subnet configured for the server address, which does not include the subnet nor the broadcast address.

PoolOffset= takes the offset of the pool from the start of subnet, or zero to use the default value.

PoolSize= takes the number of IP addresses in the pool or zero to use the default value. By default, the pool starts at the first address after the subnet address and takes up the rest of the subnet, excluding the broadcast address. If the pool includes the server address (the default), this is reserved and not handed out to clients.

DefaultLeaseTimeSec=, MaxLeaseTimeSec=

Control the default and maximum DHCP lease time to pass to clients. These settings take time values in seconds or another common time unit, depending on the suffix. The default lease time is used for clients that did not ask for a specific lease time. If a client asks for a lease time longer than the maximum lease time, it is automatically shortened to the specified time. The default lease time defaults to 1h, the maximum lease time to 12h. Shorter lease times are beneficial if the configuration data in DHCP leases changes frequently and clients shall learn the new settings with shorter latencies. Longer lease times reduce the generated DHCP network traffic.

UplinkInterface=

Specifies name or index of uplink interface, or one of the special values ":none" and ":auto". When emitting DNS, NTP, or SIP servers are enabled but no servers are specified, the servers configured in the uplink interface will be emitted. When ":auto", the link which has default gateway with higher priority will be automatically selected. When ":none", no uplink interface will be selected. Defaults to ":auto".

EmitDNS=, DNS=

EmitDNS= takes a boolean. Configures whether the DHCP leases handed out to clients shall contain DNS server information. Defaults to "yes". The DNS servers to pass to clients may be configured with the *DNS=* option, which takes a list of IPv4 addresses. If the *EmitDNS=* option is enabled but no servers configured, the servers are automatically propagated from an "uplink" interface that has appropriate servers set. The "uplink" interface is determined by the default route of the system with the highest priority. Note that this information is acquired at the time the lease is handed out, and does not take uplink interfaces into account that acquire DNS server information at a later point. If no suitable uplink interface is found the DNS server data from /etc/resolv.conf is used. Also, note that the leases are not refreshed if the uplink network configuration changes. To ensure clients regularly acquire the most current uplink DNS server information, it is thus advisable to shorten the DHCP lease time via *MaxLeaseTimeSec=* described above.

EmitNTP=, NTP=, EmitSIP=, SIP=, EmitPOP3=, POP3=, EmitSMTP=, SMTP=, EmitLPR=, LPR=

Similar to the *EmitDNS=* and *DNS=* settings described above, these settings configure whether and what server information for the indicate protocol shall be emitted as part of the DHCP lease. The same syntax, propagation semantics and defaults apply as for *EmitDNS=* and *DNS=*.

EmitRouter=

Similar to the *EmitDNS=* setting described above, this setting configures whether the DHCP lease

should contain the router option. The same syntax, propagation semantics and defaults apply as for *EmitDNS*=.

EmitTimezone=, *Timezone*=

Takes a boolean. Configures whether the DHCP leases handed out to clients shall contain timezone information. Defaults to "yes". The *Timezone*= setting takes a timezone string (such as "Europe/Berlin" or "UTC") to pass to clients. If no explicit timezone is set, the system timezone of the local host is propagated, as determined by the /etc/localtime symlink.

SendOption=

Send a raw option with value via DHCPv4 server. Takes a DHCP option number, data type and data ("option:type:value"). The option number is an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", "ipv6address", or "string". Special characters in the data string may be escaped using **C-style escapes**^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

SendVendorOption=

Send a vendor option with value via DHCPv4 server. Takes a DHCP option number, data type and data ("option:type:value"). The option number is an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using **C-style escapes**^[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

BindToInterface=

Takes a boolean value. When "yes", DHCP server socket will be bound to its network interface and all socket communication will be restricted to this interface. Defaults to "yes", except if *RelayTarget*= is used (see below), in which case it defaults to "no".

RelayTarget=

Takes an IPv4 address, which must be in the format described in **inet_pton**(3). Turns this DHCP server into a DHCP relay agent. See **RFC 1542**^[22]. The address is the address of DHCP server or another relay agent to forward DHCP messages to and from.

RelayAgentCircuitId=

Specifies value for Agent Circuit ID suboption of Relay Agent Information option. Takes a string, which must be in the format "string:value", where "value" should be replaced with the value of the suboption. Defaults to unset (means no Agent Circuit ID suboption is generated). Ignored if *RelayTarget*= is not specified.

RelayAgentRemoteId=

Specifies value for Agent Remote ID suboption of Relay Agent Information option. Takes a string, which must be in the format "string:value", where "value" should be replaced with the value of the suboption. Defaults to unset (means no Agent Remote ID suboption is generated). Ignored if *RelayTarget*= is not specified.

[DHCPSERVERSTATICLEASE] SECTION OPTIONS

The "[DHCPServerStaticLease]" section configures a static DHCP lease to assign a fixed IPv4 address to a specific device based on its MAC address. This section can be specified multiple times.

MACAddress=

The hardware address of a device to match. This key is mandatory.

Address=

The IPv4 address that should be assigned to the device that was matched with *MACAddress*=. This key is mandatory.

[IPV6SENDRA] SECTION OPTIONS

The [IPv6SendRA] section contains settings for sending IPv6 Router Advertisements and whether to act as a router, if enabled via the *IPv6SendRA*= option described above. IPv6 network prefixes or routes are defined with one or more [IPv6Prefix] or [IPv6RoutePrefix] sections.

Managed=, *OtherInformation*=

Takes a boolean. Controls whether a DHCPv6 server is used to acquire IPv6 addresses on the network link when *Managed*= is set to "true" or if only additional network information can be obtained via DHCPv6 for the network link when *OtherInformation*= is set to "true". Both settings default to "false", which means that a DHCPv6 server is not being used.

RouterLifetimeSec=

Takes a timespan. Configures the IPv6 router lifetime in seconds. When set to 0, the host is not acting as a router. Defaults to 30 minutes.

RouterPreference=

Configures IPv6 router preference if *RouterLifetimeSec*= is non-zero. Valid values are "high", "medium" and "low", with "normal" and "default" added as synonyms for "medium" just to make configuration easier. See [RFC 4191](#)^[15] for details. Defaults to "medium".

EmitDNS=, *DNS*=

DNS= specifies a list of recursive DNS server IPv6 addresses that are distributed via Router Advertisement messages when *EmitDNS*= is true. *DNS*= also takes special value "_link_local"; in that case the IPv6 link local address is distributed. If *DNS*= is empty, DNS servers are read from the [Network] section. If the [Network] section does not contain any DNS servers either, DNS servers from the uplink with the highest priority default route are used. When *EmitDNS*= is false, no DNS server information is sent in Router Advertisement messages. *EmitDNS*= defaults to true.

EmitDomains=, *Domains*=

A list of DNS search domains distributed via Router Advertisement messages when *EmitDomains*= is true. If *Domains*= is empty, DNS search domains are read from the [Network] section. If the [Network] section does not contain any DNS search domains either, DNS search domains from the uplink with the highest priority default route are used. When *EmitDomains*= is false, no DNS search domain information is sent in Router Advertisement messages. *EmitDomains*= defaults to true.

DNSLifetimeSec=

Lifetime in seconds for the DNS server addresses listed in *DNS*= and search domains listed in *Domains*=.

[IPV6PREFIX] SECTION OPTIONS

One or more [IPv6Prefix] sections contain the IPv6 prefixes that are announced via Router Advertisements. See [RFC 4861](#)^[23] for further details.

AddressAutoconfiguration=, *OnLink*=

Takes a boolean to specify whether IPv6 addresses can be autoconfigured with this prefix and whether the prefix can be used for onlink determination. Both settings default to "true" in order to ease configuration.

Prefix=

The IPv6 prefix that is to be distributed to hosts. Similarly to configuring static IPv6 addresses, the setting is configured as an IPv6 prefix and its prefix length, separated by a "/" character. Use multiple [IPv6Prefix] sections to configure multiple IPv6 prefixes since prefix lifetimes, address autoconfiguration and onlink status may differ from one prefix to another.

PreferredLifetimeSec=, *ValidLifetimeSec*=

Preferred and valid lifetimes for the prefix measured in seconds. *PreferredLifetimeSec*= defaults to 604800 seconds (one week) and *ValidLifetimeSec*= defaults to 2592000 seconds (30 days).

Assign=

Takes a boolean. When true, adds an address from the prefix. Default to false.

RouteMetric=

The metric of the prefix route. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used. This setting is ignored when *Assign*= is false.

[IPV6ROUTEPREFIX] SECTION OPTIONS

One or more [IPv6RoutePrefix] sections contain the IPv6 prefix routes that are announced via Router Advertisements. See [RFC 4191](#)^[15] for further details.

Route=

The IPv6 route that is to be distributed to hosts. Similarly to configuring static IPv6 routes, the setting is configured as an IPv6 prefix routes and its prefix route length, separated by a "/" character. Use multiple [IPv6PrefixRoutes] sections to configure multiple IPv6 prefix routes.

LifetimeSec=

Lifetime for the route prefix measured in seconds. *LifetimeSec=* defaults to 604800 seconds (one week).

[BRIDGE] SECTION OPTIONS

The [Bridge] section accepts the following keys:

UnicastFlood=

Takes a boolean. Controls whether the bridge should flood traffic for which an FDB entry is missing and the destination is unknown through this port. When unset, the kernel's default will be used.

MulticastFlood=

Takes a boolean. Controls whether the bridge should flood traffic for which an MDB entry is missing and the destination is unknown through this port. When unset, the kernel's default will be used.

MulticastToUnicast=

Takes a boolean. Multicast to unicast works on top of the multicast snooping feature of the bridge. Which means unicast copies are only delivered to hosts which are interested in it. When unset, the kernel's default will be used.

NeighborSuppression=

Takes a boolean. Configures whether ARP and ND neighbor suppression is enabled for this port. When unset, the kernel's default will be used.

Learning=

Takes a boolean. Configures whether MAC address learning is enabled for this port. When unset, the kernel's default will be used.

HairPin=

Takes a boolean. Configures whether traffic may be sent back out of the port on which it was received. When this flag is false, then the bridge will not forward traffic back out of the receiving port. When unset, the kernel's default will be used.

UseBPDU=

Takes a boolean. Configures whether STP Bridge Protocol Data Units will be processed by the bridge port. When unset, the kernel's default will be used.

FastLeave=

Takes a boolean. This flag allows the bridge to immediately stop multicast traffic on a port that receives an IGMP Leave message. It is only used with IGMP snooping if enabled on the bridge. When unset, the kernel's default will be used.

AllowPortToBeRoot=

Takes a boolean. Configures whether a given port is allowed to become a root port. Only used when STP is enabled on the bridge. When unset, the kernel's default will be used.

ProxyARP=

Takes a boolean. Configures whether proxy ARP to be enabled on this port. When unset, the kernel's default will be used.

ProxyARPWiFi=

Takes a boolean. Configures whether proxy ARP to be enabled on this port which meets extended requirements by IEEE 802.11 and Hotspot 2.0 specifications. When unset, the kernel's default will be used.

MulticastRouter=

Configures this port for having multicast routers attached. A port with a multicast router will receive all multicast traffic. Takes one of "no" to disable multicast routers on this port, "query" to let the

system detect the presence of routers, "permanent" to permanently enable multicast traffic forwarding on this port, or "temporary" to enable multicast routers temporarily on this port, not depending on incoming queries. When unset, the kernel's default will be used.

Cost=

Sets the "cost" of sending packets of this interface. Each port in a bridge may have a different speed and the cost is used to decide which link to use. Faster interfaces should have lower costs. It is an integer value between 1 and 65535.

Priority=

Sets the "priority" of sending packets on this interface. Each port in a bridge may have a different priority which is used to decide which link to use. Lower value means higher priority. It is an integer value between 0 to 63. Networkd does not set any default, meaning the kernel default value of 32 is used.

[BRIDGEFDB] SECTION OPTIONS

The [BridgeFDB] section manages the forwarding database table of a port and accepts the following keys. Specify several [BridgeFDB] sections to configure several static MAC table entries.

MACAddress=

As in the [Network] section. This key is mandatory.

Destination=

Takes an IP address of the destination VXLAN tunnel endpoint.

VLANId=

The VLAN ID for the new static MAC table entry. If omitted, no VLAN ID information is appended to the new static MAC table entry.

VNI=

The VXLAN Network Identifier (or VXLAN Segment ID) to use to connect to the remote VXLAN tunnel endpoint. Takes a number in the range 1...16777215. Defaults to unset.

AssociatedWith=

Specifies where the address is associated with. Takes one of "use", "self", "master" or "router". "use" means the address is in use. User space can use this option to indicate to the kernel that the fdb entry is in use. "self" means the address is associated with the port drivers fdb. Usually hardware. "master" means the address is associated with master devices fdb. "router" means the destination address is associated with a router. Note that it's valid if the referenced device is a VXLAN type device and has route shortcircuit enabled. Defaults to "self".

OutgoingInterface=

Specifies the name or index of the outgoing interface for the VXLAN device driver to reach the remote VXLAN tunnel endpoint. Defaults to unset.

[BRIDGEMDB] SECTION OPTIONS

The [BridgeMDB] section manages the multicast membership entries forwarding database table of a port and accepts the following keys. Specify several [BridgeMDB] sections to configure several permanent multicast membership entries.

MulticastGroupAddress=

Specifies the IPv4 or IPv6 multicast group address to add. This setting is mandatory.

VLANId=

The VLAN ID for the new entry. Valid ranges are 0 (no VLAN) to 4094. Optional, defaults to 0.

[LLDP] SECTION OPTIONS

The [LLDP] section manages the Link Layer Discovery Protocol (LLDP) and accepts the following keys:

MUDURL=

When configured, the specified Manufacturer Usage Descriptions (MUD) URL will be sent in LLDP packets. The syntax and semantics are the same as for *MUDURL*= in the [DHCPv4] section described above.

The MUD URLs received via LLDP packets are saved and can be read using the `sd_lldp_neighbor_get_mud_url()` function.

[CAN] SECTION OPTIONS

The [CAN] section manages the Controller Area Network (CAN bus) and accepts the following keys:

BitRate=

The bitrate of CAN device in bits per second. The usual SI prefixes (K, M) with the base of 1000 can be used here. Takes a number in the range 1...4294967295.

SamplePoint=

Optional sample point in percent with one decimal (e.g. "75%", "87.5%") or permille (e.g. "875‰").

DataBitRate=, *DataSamplePoint*=

The bitrate and sample point for the data phase, if CAN-FD is used. These settings are analogous to the *BitRate*= and *SamplePoint*= keys.

FDMode=

Takes a boolean. When "yes", CAN-FD mode is enabled for the interface. Note, that a bitrate and optional sample point should also be set for the CAN-FD data phase using the *DataBitRate*= and *DataSamplePoint*= keys.

FDNonISO=

Takes a boolean. When "yes", non-ISO CAN-FD mode is enabled for the interface. When unset, the kernel's default will be used.

RestartSec=

Automatic restart delay time. If set to a non-zero value, a restart of the CAN controller will be triggered automatically in case of a bus-off condition after the specified delay time. Subsecond delays can be specified using decimals (e.g. "0.1s") or a "ms" or "us" postfix. Using "infinity" or "0" will turn the automatic restart off. By default automatic restart is disabled.

Termination=

Takes a boolean. When "yes", the termination resistor will be selected for the bias network. When unset, the kernel's default will be used.

TripleSampling=

Takes a boolean. When "yes", three samples (instead of one) are used to determine the value of a received bit by majority rule. When unset, the kernel's default will be used.

BusErrorReporting=

Takes a boolean. When "yes", reporting of CAN bus errors is activated (those include single bit, frame format, and bit stuffing errors, unable to send dominant bit, unable to send recessive bit, bus overload, active error announcement, error occurred on transmission). When unset, the kernel's default will be used. Note: in case of a CAN bus with a single CAN device, sending a CAN frame may result in a huge number of CAN bus errors.

ListenOnly=

Takes a boolean. When "yes", listen-only mode is enabled. When the interface is in listen-only mode, the interface neither transmit CAN frames nor send ACK bit. Listen-only mode is important to debug CAN networks without interfering with the communication or acknowledge the CAN frame. When unset, the kernel's default will be used.

[QDISC] SECTION OPTIONS

The [QDisc] section manages the traffic control queueing discipline (qdisc).

Parent=

Specifies the parent Queueing Discipline (qdisc). Takes one of "clsact" or "ingress". This is mandatory.

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[NETWORKEMULATOR] SECTION OPTIONS

The [NetworkEmulator] section manages the queueing discipline (qdisc) of the network emulator. It can be used to configure the kernel packet scheduler and simulate packet delay and loss for UDP or TCP applications, or limit the bandwidth usage of a particular service to simulate internet connections.

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

DelaySec=

Specifies the fixed amount of delay to be added to all packets going out of the interface. Defaults to unset.

DelayJitterSec=

Specifies the chosen delay to be added to the packets outgoing to the network interface. Defaults to unset.

PacketLimit=

Specifies the maximum number of packets the qdisc may hold queued at a time. An unsigned integer in the range 0–4294967294. Defaults to 1000.

LossRate=

Specifies an independent loss probability to be added to the packets outgoing from the network interface. Takes a percentage value, suffixed with "%". Defaults to unset.

DuplicateRate=

Specifies that the chosen percent of packets is duplicated before queuing them. Takes a percentage value, suffixed with "%". Defaults to unset.

[TOKENBUCKETFILTER] SECTION OPTIONS

The [TokenBucketFilter] section manages the queueing discipline (qdisc) of token bucket filter (tbf).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

LatencySec=

Specifies the latency parameter, which specifies the maximum amount of time a packet can sit in the Token Bucket Filter (TBF). Defaults to unset.

LimitBytes=

Takes the number of bytes that can be queued waiting for tokens to become available. When the size is suffixed with K, M, or G, it is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

BurstBytes=

Specifies the size of the bucket. This is the maximum amount of bytes that tokens can be available for instantaneous transfer. When the size is suffixed with K, M, or G, it is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

Rate=

Specifies the device specific bandwidth. When suffixed with K, M, or G, the specified bandwidth is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset.

MPUBytes=

The Minimum Packet Unit (MPU) determines the minimal token usage (specified in bytes) for a packet. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to zero.

PeakRate=

Takes the maximum depletion rate of the bucket. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset.

MTUBytes=

Specifies the size of the peakrate bucket. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

[PIE] SECTION OPTIONS

The [PIE] section manages the queueing discipline (qdisc) of Proportional Integral controller–Enhanced (PIE).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 1...4294967294. Defaults to unset and kernel's default is used.

[FLOWQUEUEPIE] SECTION OPTIONS

The "[FlowQueuePIE]" section manages the queueing discipline (qdisc) of Flow Queue Proportional Integral controller–Enhanced (fq_pie).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer ranges 1 to 4294967294. Defaults to unset and kernel's default is used.

[STOCHASTICFAIRBLUE] SECTION OPTIONS

The [StochasticFairBlue] section manages the queueing discipline (qdisc) of stochastic fair blue (sfb).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming

packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[STOCHASTICFAIRNESSQUEUEING] SECTION OPTIONS

The [StochasticFairnessQueueing] section manages the queueing discipline (qdisc) of stochastic fairness queueing (sfq).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PerturbPeriodSec=

Specifies the interval in seconds for queue algorithm perturbation. Defaults to unset.

[BFIFO] SECTION OPTIONS

The [BFIFO] section manages the queueing discipline (qdisc) of Byte limited Packet First In First Out (bfifo).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

LimitBytes=

Specifies the hard limit in bytes on the FIFO buffer size. The size limit prevents overflow in case the kernel is unable to dequeue packets as quickly as it receives them. When this limit is reached, incoming packets are dropped. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel default is used.

[PFIFO] SECTION OPTIONS

The [PFIFO] section manages the queueing discipline (qdisc) of Packet First In First Out (pfifo).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the number of packets in the FIFO queue. The size limit prevents overflow in case the kernel is unable to dequeue packets as quickly as it receives them. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[PFIFOHEADDROP] SECTION OPTIONS

The [PFIFOHeadDrop] section manages the queueing discipline (qdisc) of Packet First In First Out Head Drop (pfifo_head_drop).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class

identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

As in [PFIFO] section.

[PFIFOFAST] SECTION OPTIONS

The [PFIFOFast] section manages the queueing discipline (qdisc) of Packet First In First Out Fast (pfifo_fast).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[CAKE] SECTION OPTIONS

The [CAKE] section manages the queueing discipline (qdisc) of Common Applications Kept Enhanced (CAKE).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

OverheadBytes=

Specifies that bytes to be added to the size of each packet. Bytes may be negative. Takes an integer in the range from -64 to 256. Defaults to unset and kernel's default is used.

Bandwidth=

Specifies the shaper bandwidth. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset and kernel's default is used.

[CONTROLLEDDELAY] SECTION OPTIONS

The [ControlledDelay] section manages the queueing discipline (qdisc) of controlled delay (CoDel).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

TargetSec=

Takes a timespan. Specifies the acceptable minimum standing/persistent queue delay. Defaults to unset and kernel's default is used.

IntervalSec=

Takes a timespan. This is used to ensure that the measured minimum delay does not become too stale. Defaults to unset and kernel's default is used.

ECN=

Takes a boolean. This can be used to mark packets instead of dropping them. Defaults to unset and kernel's default is used.

CEThresholdSec=

Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[DEFICITROUNDROBINSCHEDULER] SECTION OPTIONS

The [DeficitRoundRobinScheduler] section manages the queueing discipline (qdisc) of Deficit Round Robin Scheduler (DRR).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[DEFICITROUNDROBINSCHEDULERCLASS] SECTION OPTIONS

The [DeficitRoundRobinSchedulerClass] section manages the traffic control class of Deficit Round Robin Scheduler (DRR).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=

Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to unset.

QuantumBytes=

Specifies the amount of bytes a flow is allowed to dequeue before the scheduler moves to the next class. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to the MTU of the interface.

[ENHANCEDTRANSMISSIONSELECTION] SECTION OPTIONS

The [EnhancedTransmissionSelection] section manages the queueing discipline (qdisc) of Enhanced Transmission Selection (ETS).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

Bands=

Specifies the number of bands. An unsigned integer in the range 1–16. This value has to be at least large enough to cover the strict bands specified through the *StrictBands*= and bandwidth-sharing bands specified in *QuantumBytes*=.

StrictBands=

Specifies the number of bands that should be created in strict mode. An unsigned integer in the range 1–16.

QuantumBytes=

Specifies the white-space separated list of quantum used in band-sharing bands. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

PriorityMap=

The priority map maps the priority of a packet to a band. The argument is a whitespace separated list of numbers. The first number indicates which band the packets with priority 0 should be put to, the second is for priority 1, and so on. There can be up to 16 numbers in the list. If there are fewer, the default band that traffic with one of the unmentioned priorities goes to is the last one. Each band number must be in the range 0...255. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

[GENERICRANDOMEARLYDETECTION] SECTION OPTIONS

The [GenericRandomEarlyDetection] section manages the queueing discipline (qdisc) of Generic Random Early Detection (GRED).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

VirtualQueues=

Specifies the number of virtual queues. Takes an integer in the range 1...16. Defaults to unset and kernel's default is used.

DefaultVirtualQueue=

Specifies the number of default virtual queue. This must be less than *VirtualQueue=*. Defaults to unset and kernel's default is used.

GenericRIO=

Takes a boolean. It turns on the RIO-like buffering scheme. Defaults to unset and kernel's default is used.

[FAIRQUEUEINGCONTROLLEDDELAY] SECTION OPTIONS

The [FairQueueingControlledDelay] section manages the queueing discipline (qdisc) of fair queuing controlled delay (FQ-CoDel).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Defaults to unset and kernel's default is used.

MemoryLimitBytes=

Specifies the limit on the total number of bytes that can be queued in this FQ-CoDel instance. When

suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

Flows=

Specifies the number of flows into which the incoming packets are classified. Defaults to unset and kernel's default is used.

TargetSec=

Takes a timespan. Specifies the acceptable minimum standing/persistent queue delay. Defaults to unset and kernel's default is used.

IntervalSec=

Takes a timespan. This is used to ensure that the measured minimum delay does not become too stale. Defaults to unset and kernel's default is used.

QuantumBytes=

Specifies the number of bytes used as the "deficit" in the fair queuing algorithm timespan. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

ECN=

Takes a boolean. This can be used to mark packets instead of dropping them. Defaults to unset and kernel's default is used.

CEThresholdSec=

Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[FAIRQUEUEING] SECTION OPTIONS

The [FairQueueing] section manages the queueing discipline (qdisc) of fair queue traffic policing (FQ).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Defaults to unset and kernel's default is used.

FlowLimit=

Specifies the hard limit on the maximum number of packets queued per flow. Defaults to unset and kernel's default is used.

QuantumBytes=

Specifies the credit per dequeue RR round, i.e. the amount of bytes a flow is allowed to dequeue at once. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

InitialQuantumBytes=

Specifies the initial sending rate credit, i.e. the amount of bytes a new flow is allowed to dequeue initially. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

MaximumRate=

Specifies the maximum sending rate of a flow. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset and kernel's default is used.

Buckets=

Specifies the size of the hash table used for flow lookups. Defaults to unset and kernel's default is used.

OrphanMask=

Takes an unsigned integer. For packets not owned by a socket, fq is able to mask a part of hash and reduce number of buckets associated with the traffic. Defaults to unset and kernel's default is used.

Pacing=

Takes a boolean, and enables or disables flow pacing. Defaults to unset and kernel's default is used.

CEThresholdSec=

Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[TRIVIALLINKEQUALIZER] SECTION OPTIONS

The [TrivialLinkEqualizer] section manages the queueing discipline (qdisc) of trivial link equalizer (teql).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

Id=

Specifies the interface ID "N" of teql. Defaults to "0". Note that when teql is used, currently, the module **sch_teql** with **max_equalizers=N+1** option must be loaded before **systemd-networkd** is started.

[HIERARCHYTOKENBUCKET] SECTION OPTIONS

The [HierarchyTokenBucket] section manages the queueing discipline (qdisc) of hierarchy token bucket (htb).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

DefaultClass=

Takes the minor id in hexadecimal of the default class. Unclassified traffic gets sent to the class. Defaults to unset.

RateToQuantum=

Takes an unsigned integer. The DRR quantums are calculated by dividing the value configured in **Rate=** by **RateToQuantum=**.

[HIERARCHYTOKENBUCKETCLASS] SECTION OPTIONS

The [HierarchyTokenBucketClass] section manages the traffic control class of hierarchy token bucket (htb).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=

Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to unset.

Priority=

Specifies the priority of the class. In the round-robin process, classes with the lowest priority field are tried for packets first.

QuantumBytes=

Specifies how many bytes to serve from leaf at once. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

MTUBytes=

Specifies the maximum packet size we create. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

OverheadBytes=

Takes an unsigned integer which specifies per-packet size overhead used in rate computations. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

Rate=

Specifies the maximum rate this class and all its children are guaranteed. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. This setting is mandatory.

CeilRate=

Specifies the maximum rate at which a class can send, if its parent has bandwidth to spare. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. When unset, the value specified with *Rate*= is used.

BufferBytes=

Specifies the maximum bytes burst which can be accumulated during idle period. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

CeilBufferBytes=

Specifies the maximum bytes burst for ceil which can be accumulated during idle period. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

[HEAVYHITTERFILTER] SECTION OPTIONS

The [HeavyHitterFilter] section manages the queueing discipline (qdisc) of Heavy Hitter Filter (hhf).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[QUICKFAIRQUEUEING] SECTION OPTIONS

The [QuickFairQueueing] section manages the queueing discipline (qdisc) of Quick Fair Queueing (QFQ).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[QUICKFAIRQUEUEINGCLASS] SECTION OPTIONS

The [QuickFairQueueingClass] section manages the traffic control class of Quick Fair Queueing (qfq).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=

Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–0xffff separated with a colon ("major:minor"). Defaults to unset.

Weight=

Specifies the weight of the class. Takes an integer in the range 1...1023. Defaults to unset in which case the kernel default is used.

MaxPacketBytes=

Specifies the maximum packet size in bytes for the class. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. When unset, the kernel default is used.

[BRIDGEVLAN] SECTION OPTIONS

The [BridgeVLAN] section manages the VLAN ID configuration of a bridge port and accepts the following keys. Specify several [BridgeVLAN] sections to configure several VLAN entries. The *VLANFiltering*= option has to be enabled, see the [Bridge] section in **systemd.netdev(5)**.

VLAN=

The VLAN ID allowed on the port. This can be either a single ID or a range M–N. VLAN IDs are valid from 1 to 4094.

EgressUntagged=

The VLAN ID specified here will be used to untag frames on egress. Configuring *EgressUntagged*= implicates the use of *VLAN*= above and will enable the VLAN ID for ingress as well. This can be either a single ID or a range M–N.

PVID=

The Port VLAN ID specified here is assigned to all untagged frames at ingress. *PVID*= can be used only once. Configuring *PVID*= implicates the use of *VLAN*= above and will enable the VLAN ID for ingress as well.

EXAMPLES

Example 1. Static network configuration

```
# /etc/systemd/network/50-static.network
[Match]
Name=enp2s0

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

This brings interface "enp2s0" up with a static address. The specified gateway will be used for a default route.

Example 2. DHCP on ethernet links

```
# /etc/systemd/network/80-dhcp.network
[Match]
Name=en*
```

```
[Network]
DHCP=yes
```

This will enable DHCPv4 and DHCPv6 on all interfaces with names starting with "en" (i.e. ethernet interfaces).

Example 3. IPv6 Prefix Delegation

```
# /etc/systemd/network/55-ipv6-pd-upstream.network
[Match]
Name=enp1s0
```

```
[Network]
DHCP=ipv6
```

```
# /etc/systemd/network/56-ipv6-pd-downstream.network
[Match]
Name=enp2s0
```

```
[Network]
IPv6SendRA=yes
DHCPv6PrefixDelegation=yes
```

This will enable DHCPv6-PD on the interface enp1s0 as an upstream interface where the DHCPv6 client is running and enp2s0 as a downstream interface where the prefix is delegated to. The delegated prefixes are distributed by IPv6 Router Advertisement on the downstream network.

Example 4. A bridge with two enslaved links

```
# /etc/systemd/network/25-bridge-static.network
[Match]
Name=bridge0
```

```
[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
DNS=192.168.0.1
```

```
# /etc/systemd/network/25-bridge-slave-interface-1.network
[Match]
Name=enp2s0
```

```
[Network]
Bridge=bridge0
```

```
# /etc/systemd/network/25-bridge-slave-interface-2.network
[Match]
Name=wlp3s0
```

```
[Network]
Bridge=bridge0
```

This creates a bridge and attaches devices "enp2s0" and "wlp3s0" to it. The bridge will have the specified static address and network assigned, and a default route via the specified gateway will be added. The specified DNS server will be added to the global list of DNS resolvers.

Example 5. Bridge port with VLAN forwarding

```
# /etc/systemd/network/25-bridge-slave-interface-1.network
[Match]
Name=enp2s0

[Network]
Bridge=bridge0

[BridgeVLAN]
VLAN=1-32
PVID=42
EgressUntagged=42

[BridgeVLAN]
VLAN=100-200

[BridgeVLAN]
EgressUntagged=300-400
```

This overrides the configuration specified in the previous example for the interface "enp2s0", and enables VLAN on that bridge port. VLAN IDs 1–32, 42, 100–400 will be allowed. Packets tagged with VLAN IDs 42, 300–400 will be untagged when they leave on this interface. Untagged packets which arrive on this interface will be assigned VLAN ID 42.

Example 6. Various tunnels

```
/etc/systemd/network/25-tunnels.network
[Match]
Name=ens1
```

```
[Network]
Tunnel=ipip-tun
Tunnel=sit-tun
Tunnel=gre-tun
Tunnel=vti-tun
```

```
/etc/systemd/network/25-tunnel-ipip.netdev
[NetDev]
Name=ipip-tun
Kind=ipip
```

```
/etc/systemd/network/25-tunnel-sit.netdev
[NetDev]
Name=sit-tun
Kind=sit
```

```
/etc/systemd/network/25-tunnel-gre.netdev
[NetDev]
Name=gre-tun
Kind=gre
```

```
/etc/systemd/network/25-tunnel-vti.netdev
[NetDev]
```

```
Name=vti-tun
Kind=vti
```

This will bring interface "ens1" up and create an IPIP tunnel, a SIT tunnel, a GRE tunnel, and a VTI tunnel using it.

Example 7. A bond device

```
# /etc/systemd/network/30-bond1.network
[Match]
Name=bond1

[Network]
DHCP=ipv6

# /etc/systemd/network/30-bond1.netdev
[NetDev]
Name=bond1
Kind=bond

# /etc/systemd/network/30-bond1-dev1.network
[Match]
MACAddress=52:54:00:e9:64:41

[Network]
Bond=bond1

# /etc/systemd/network/30-bond1-dev2.network
[Match]
MACAddress=52:54:00:e9:64:42

[Network]
Bond=bond1
```

This will create a bond device "bond1" and enslave the two devices with MAC addresses 52:54:00:e9:64:41 and 52:54:00:e9:64:42 to it. IPv6 DHCP will be used to acquire an address.

Example 8. Virtual Routing and Forwarding (VRF)

Add the "bond1" interface to the VRF master interface "vrf1". This will redirect routes generated on this interface to be within the routing table defined during VRF creation. For kernels before 4.8 traffic won't be redirected towards the VRFs routing table unless specific ip-rules are added.

```
# /etc/systemd/network/25-vrf.network
[Match]
Name=bond1

[Network]
VRF=vrf1
```

Example 9. MacVTap

This brings up a network interface "macvtap-test" and attaches it to "enp0s25".

```
# /lib/systemd/network/25-macvtap.network
[Match]
Name=enp0s25
```

```
[Network]
MACVTAP=macvtap-test
```

Example 10. A Xfrm interface with physical underlying device.

```
# /etc/systemd/network/27-xfrm.netdev
```

```
[NetDev]
Name=xfrm0
Kind=xfrm
```

```
[Xfrm]
InterfaceId=7
```

```
# /etc/systemd/network/27-eth0.network
```

```
[Match]
Name=eth0
```

```
[Network]
Xfrm=xfrm0
```

This creates a "xfrm0" interface and binds it to the "eth0" device. This allows hardware based ipsec offloading to the "eth0" nic. If offloading is not needed, xfrm interfaces can be assigned to the "lo" device.

SEE ALSO

systemd(1), systemd-networkd.service(8), systemd.link(5), systemd.netdev(5), systemd-resolved.service(8)

NOTES

1. RFC 7217
<https://tools.ietf.org/html/rfc7217>
2. Link-Local Multicast Name Resolution
<https://tools.ietf.org/html/rfc4795>
3. Multicast DNS
<https://tools.ietf.org/html/rfc6762>
4. DNS-over-TLS
<https://tools.ietf.org/html/rfc7858>
5. DNSSEC
<https://tools.ietf.org/html/rfc4033>
6. IEEE 802.1AB-2016
<https://standards.ieee.org/findstds/standard/802.1AB-2016.html>
7. ip-sysctl.txt
<https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt>
8. RFC 4941
<https://tools.ietf.org/html/rfc4941>
9. RFC 1027
<https://tools.ietf.org/html/rfc1027>
10. RFC 6275
<https://tools.ietf.org/html/rfc6275>
11. RFC 5227
<https://tools.ietf.org/html/rfc5227>
12. RFC 4862
<https://tools.ietf.org/html/rfc4862>

13. RFC 3041
<https://tools.ietf.org/html/rfc3041>
14. RFC 3484
<https://tools.ietf.org/html/rfc3484>
15. RFC 4191
<https://tools.ietf.org/html/rfc4191>
16. RFC 8520
<https://tools.ietf.org/html/rfc8520>
17. RFC 7844
<https://tools.ietf.org/html/rfc7844>
18. C-style escapes
https://en.wikipedia.org/wiki/Escape_sequences_in_C#Table_of_escape_sequences
19. RFC 7084
<https://tools.ietf.org/html/rfc7084>
20. RFC 3315
<https://tools.ietf.org/html/rfc3315#section-17.2.1>
21. RFC 4291
<https://tools.ietf.org/html/rfc4291#section-2.5.4>
22. RFC 1542
<https://tools.ietf.org/html/rfc1542>
23. RFC 4861
<https://tools.ietf.org/html/rfc4861>

NAME

systemd.service – Service unit configuration

SYNOPSIS

service.service

DESCRIPTION

A unit configuration file whose name ends in ".service" encodes information about a process controlled and supervised by systemd.

This man page lists the configuration options specific to this unit type. See **systemd.unit(5)** for the common options of all unit configuration files. The common configuration items are configured in the generic [Unit] and [Install] sections. The service specific configuration options are configured in the [Service] section.

Additional options are listed in **systemd.exec(5)**, which define the execution environment the commands are executed in, and in **systemd.kill(5)**, which define the way the processes of the service are terminated, and in **systemd.resource-control(5)**, which configure resource control settings for the processes of the service.

If a service is requested under a certain name but no unit configuration file is found, systemd looks for a SysV init script by the same name (with the .service suffix removed) and dynamically creates a service unit from that script. This is useful for compatibility with SysV. Note that this compatibility is quite comprehensive but not 100%. For details about the incompatibilities, see the [Incompatibilities with SysV^{\[1\]}](#) document.

The **systemd-run(1)** command allows creating .service and .scope units dynamically and transiently from the command line.

SERVICE TEMPLATES

It is possible for **systemd** services to take a single argument via the "*service@argument.service*" syntax. Such services are called "instantiated" services, while the unit definition without the *argument* parameter is called a "template". An example could be a dhcpcd@.service service template which takes a network interface as a parameter to form an instantiated service. Within the service file, this parameter or "instance name" can be accessed with %–specifiers. See **systemd.unit(5)** for details.

AUTOMATIC DEPENDENCIES

Implicit Dependencies

The following dependencies are implicitly added:

- Services with *Type=dbus* set automatically acquire dependencies of type *Requires=* and *After=* on *dbus.socket*.
- Socket activated services are automatically ordered after their activating .socket units via an automatic *After=* dependency. Services also pull in all .socket units listed in *Sockets=* via automatic *Wants=* and *After=* dependencies.

Additional implicit dependencies may be added as result of execution and resource control parameters as documented in **systemd.exec(5)** and **systemd.resource-control(5)**.

Default Dependencies

The following dependencies are added unless *DefaultDependencies=no* is set:

- Service units will have dependencies of type *Requires=* and *After=* on *sysinit.target*, a dependency of type *After=* on *basic.target* as well as dependencies of type *Conflicts=* and *Before=* on *shutdown.target*. These ensure that normal service units pull in basic system initialization, and are terminated cleanly prior to system shutdown. Only services involved with early boot or late system shutdown should disable this option.
- Instanced service units (i.e. service units with an "@" in their name) are assigned by default a per-template slice unit (see **systemd.slice(5)**), named after the template unit, containing all instances of the specific template. This slice is normally stopped at shutdown, together with all template instances. If that is not desired, set *DefaultDependencies=no* in the template unit, and

either define your own per-template slice unit file that also sets *DefaultDependencies=no*, or set *Slice=system.slice* (or another suitable slice) in the template unit. Also see **systemd.resource-control(5)**.

OPTIONS

Service files must include a [Service] section, which carries information about the service and the process it supervises. A number of options that may be used in this section are shared with other unit types. These options are documented in **systemd.exec(5)**, **systemd.kill(5)** and **systemd.resource-control(5)**. The options specific to the [Service] section of service units are the following:

Type=

Configures the process start-up type for this service unit. One of **simple**, **exec**, **forking**, **oneshot**, **dbus**, **notify** or **idle**:

- If set to **simple** (the default if *ExecStart=* is specified but neither *Type=* nor *BusName=* are), the service manager will consider the unit started immediately after the main service process has been forked off. It is expected that the process configured with *ExecStart=* is the main process of the service. In this mode, if the process offers functionality to other processes on the system, its communication channels should be installed before the service is started up (e.g. sockets set up by systemd, via socket activation), as the service manager will immediately proceed starting follow-up units, right after creating the main service process, and before executing the service's binary. Note that this means **systemctl start** command lines for **simple** services will report success even if the service's binary cannot be invoked successfully (for example because the selected *User=* doesn't exist, or the service binary is missing).
- The **exec** type is similar to **simple**, but the service manager will consider the unit started immediately after the main service binary has been executed. The service manager will delay starting of follow-up units until that point. (Or in other words: **simple** proceeds with further jobs right after **fork()** returns, while **exec** will not proceed before both **fork()** and **execve()** in the service process succeeded.) Note that this means **systemctl start** command lines for **exec** services will report failure when the service's binary cannot be invoked successfully (for example because the selected *User=* doesn't exist, or the service binary is missing).
- If set to **forking**, it is expected that the process configured with *ExecStart=* will call **fork()** as part of its start-up. The parent process is expected to exit when start-up is complete and all communication channels are set up. The child continues to run as the main service process, and the service manager will consider the unit started when the parent process exits. This is the behavior of traditional UNIX services. If this setting is used, it is recommended to also use the *PIDFile=* option, so that systemd can reliably identify the main process of the service. systemd will proceed with starting follow-up units as soon as the parent process exits.
- Behavior of **oneshot** is similar to **simple**; however, the service manager will consider the unit up after the main process exits. It will then start follow-up units. *RemainAfterExit=* is particularly useful for this type of service. *Type=oneshot* is the implied default if neither *Type=* nor *ExecStart=* are specified. Note that if this option is used without *RemainAfterExit=* the service will never enter "active" unit state, but directly transition from "activating" to "deactivating" or "dead" since no process is configured that shall run continuously. In particular this means that after a service of this type ran (and which has *RemainAfterExit=* not set) it will not show up as started afterwards, but as dead.
- Behavior of **dbus** is similar to **simple**; however, it is expected that the service acquires a name on the D-Bus bus, as configured by *BusName=*. systemd will proceed with starting follow-up units after the D-Bus bus name has been acquired. Service units with this option configured implicitly gain dependencies on the **dbus.socket** unit. This type is the default if *BusName=* is specified. A service unit of this type is considered to be in the activating state until the specified bus name is acquired. It is considered activated while the bus name is taken. Once the bus name is released the service is considered being no longer functional which has the effect that the service manager attempts to terminate any remaining processes belonging to the service. Services that drop their bus name as part of their shutdown logic thus should be

prepared to receive a **SIGTERM** (or whichever signal is configured in *KillSignal*=) as result.

- Behavior of **notify** is similar to **exec**; however, it is expected that the service sends a notification message via **sd_notify**(3) or an equivalent call when it has finished starting up. systemd will proceed with starting follow-up units after this notification message has been sent. If this option is used, *NotifyAccess*= (see below) should be set to open access to the notification socket provided by systemd. If *NotifyAccess*= is missing or set to **none**, it will be forcibly set to **main**.
- Behavior of **idle** is very similar to **simple**; however, actual execution of the service program is delayed until all active jobs are dispatched. This may be used to avoid interleaving of output of shell services with the status output on the console. Note that this type is useful only to improve console output, it is not useful as a general unit ordering tool, and the effect of this service type is subject to a 5s timeout, after which the service program is invoked anyway.

It is generally recommended to use *Type=simple* for long-running services whenever possible, as it is the simplest and fastest option. However, as this service type won't propagate service start-up failures and doesn't allow ordering of other units against completion of initialization of the service (which for example is useful if clients need to connect to the service through some form of IPC, and the IPC channel is only established by the service itself — in contrast to doing this ahead of time through socket or bus activation or similar), it might not be sufficient for many cases. If so, **notify** or **dbus** (the latter only in case the service provides a D-Bus interface) are the preferred options as they allow service program code to precisely schedule when to consider the service started up successfully and when to proceed with follow-up units. The **notify** service type requires explicit support in the service codebase (as **sd_notify()** or an equivalent API needs to be invoked by the service at the appropriate time) — if it's not supported, then **forking** is an alternative: it supports the traditional UNIX service start-up protocol. Finally, **exec** might be an option for cases where it is enough to ensure the service binary is invoked, and where the service binary itself executes no or little initialization on its own (and its initialization is unlikely to fail). Note that using any type other than **simple** possibly delays the boot process, as the service manager needs to wait for service initialization to complete. It is hence recommended not to needlessly use any types other than **simple**. (Also note it is generally not recommended to use **idle** or **oneshot** for long-running services.)

RemainAfterExit=

Takes a boolean value that specifies whether the service shall be considered active even when all its processes exited. Defaults to **no**.

GuessMainPID=

Takes a boolean value that specifies whether systemd should try to guess the main PID of a service if it cannot be determined reliably. This option is ignored unless **Type=forking** is set and **PIDFile**= is unset because for the other types or with an explicitly configured PID file, the main PID is always known. The guessing algorithm might come to incorrect conclusions if a daemon consists of more than one process. If the main PID cannot be determined, failure detection and automatic restarting of a service will not work reliably. Defaults to **yes**.

PIDFile=

Takes a path referring to the PID file of the service. Usage of this option is recommended for services where *Type*= is set to **forking**. The path specified typically points to a file below /run/. If a relative path is specified it is hence prefixed with /run/. The service manager will read the PID of the main process of the service from this file after start-up of the service. The service manager will not write to the file configured here, although it will remove the file after the service has shut down if it still exists. The PID file does not need to be owned by a privileged user, but if it is owned by an unprivileged user additional safety restrictions are enforced: the file may not be a symlink to a file owned by a different user (neither directly nor indirectly), and the PID file must refer to a process already belonging to the service.

Note that PID files should be avoided in modern projects. Use **Type=notify** or **Type=simple** where

possible, which does not require use of PID files to determine the main process of a service and avoids needless forking.

BusName=

Takes a D-Bus destination name that this service shall use. This option is mandatory for services where *Type*= is set to **dbus**. It is recommended to always set this property if known to make it easy to map the service name to the D-Bus destination. In particular, **systemctl** **service-log-level**/**service-log-target** verbs make use of this.

ExecStart=

Commands with their arguments that are executed when this service is started. The value is split into zero or more command lines according to the rules described below (see section "Command Lines" below).

Unless *Type*= is **oneshot**, exactly one command must be given. When *Type=oneshot* is used, zero or more commands may be specified. Commands may be specified by providing multiple command lines in the same directive, or alternatively, this directive may be specified more than once with the same effect. If the empty string is assigned to this option, the list of commands to start is reset, prior assignments of this option will have no effect. If no *ExecStart*= is specified, then the service must have *RemainAfterExit=yes* and at least one *ExecStop*= line set. (Services lacking both *ExecStart*= and *ExecStop*= are not valid.)

For each of the specified commands, the first argument must be either an absolute path to an executable or a simple file name without any slashes. Optionally, this filename may be prefixed with a number of special characters:

Table 1. Special executable prefixes

"@", "-", ":" , and one of "+"|"!"|"!!" may be used together and they can appear in any order. However, only one of "+", "!", "!!" may be used at a time. Note that these prefixes are also supported for the other command line settings, i.e. *ExecStartPre=*, *ExecStartPost=*, *ExecReload=*, *ExecStop=* and *ExecStopPost=*.

If more than one command is specified, the commands are invoked sequentially in the order they appear in the unit file. If one of the commands fails (and is not prefixed with "-"), other lines are not executed, and the unit is considered failed.

Unless *Type=forking* is set, the process started via this command line will be considered the main process of the daemon.

ExecStartPre=, *ExecStartPost=*

Additional commands that are executed before or after the command in *ExecStart=*, respectively. Syntax is the same as for *ExecStart=*, except that multiple command lines are allowed and the commands are executed one after the other, serially.

If any of those commands (not prefixed with "-") fail, the rest are not executed and the unit is considered failed.

ExecStart= commands are only run after all *ExecStartPre=* commands that were not prefixed with a "-" exit successfully.

ExecStartPost= commands are only run after the commands specified in *ExecStart=* have been invoked successfully, as determined by *Type* (i.e. the process has been started for *Type=simple* or *Type=idle*, the last *ExecStart=* process exited successfully for *Type=oneshot*, the initial process exited successfully for *Type=forking*, "READY=1" is sent for *Type=notify*, or the *BusName=* has been taken for *Type=dbus*).

Note that *ExecStartPre=* may not be used to start long-running processes. All processes forked off by processes invoked via *ExecStartPre=* will be killed before the next service process is run.

Note that if any of the commands specified in *ExecStartPre=*, *ExecStart=*, or *ExecStartPost=* fail (and are not prefixed with "-", see above) or time out before the service is fully up, execution continues with commands specified in *ExecStopPost=*, the commands in *ExecStop=* are skipped.

Note that the execution of *ExecStartPost=* is taken into account for the purpose of *Before=/After=* ordering constraints.

ExecCondition=

Optional commands that are executed before the command(s) in *ExecStartPre=*. Syntax is the same as for *ExecStart=*, except that multiple command lines are allowed and the commands are executed one after the other, serially.

The behavior is like an *ExecStartPre=* and condition check hybrid: when an *ExecCondition=* command exits with exit code 1 through 254 (inclusive), the remaining commands are skipped and the unit is *not* marked as failed. However, if an *ExecCondition=* command exits with 255 or abnormally (e.g. timeout, killed by a signal, etc.), the unit will be considered failed (and remaining commands will be skipped). Exit code of 0 or those matching *SuccessExitStatus=* will continue execution to the next command(s).

The same recommendations about not running long-running processes in *ExecStartPre=* also applies to *ExecCondition=*. *ExecCondition=* will also run the commands in *ExecStopPost=*, as part of stopping the service, in the case of any non-zero or abnormal exits, like the ones described above.

ExecReload=

Commands to execute to trigger a configuration reload in the service. This argument takes multiple command lines, following the same scheme as described for *ExecStart=* above. Use of this setting is optional. Specifier and environment variable substitution is supported here following the same scheme as for *ExecStart=*.

One additional, special environment variable is set: if known, *\$MAINPID* is set to the main process of the daemon, and may be used for command lines like the following:

```
ExecReload=kill -HUP $MAINPID
```

Note however that reloading a daemon by sending a signal (as with the example line above) is usually not a good choice, because this is an asynchronous operation and hence not suitable to order reloads of multiple services against each other. It is strongly recommended to set *ExecReload=* to a command that not only triggers a configuration reload of the daemon, but also synchronously waits for it to complete. For example, **dbus-broker**(1) uses the following:

```
ExecReload=busctl call org.freedesktop.DBus \
    /org/freedesktop/DBus org.freedesktop.DBus \
    ReloadConfig
```

ExecStop=

Commands to execute to stop the service started via *ExecStart=*. This argument takes multiple command lines, following the same scheme as described for *ExecStart=* above. Use of this setting is optional. After the commands configured in this option are run, it is implied that the service is stopped, and any processes remaining for it are terminated according to the *KillMode=* setting (see **systemd.kill**(5)). If this option is not specified, the process is terminated by sending the signal specified in *KillSignal=* or *RestartKillSignal=* when service stop is requested. Specifier and environment variable substitution is supported (including *\$MAINPID*, see above).

Note that it is usually not sufficient to specify a command for this setting that only asks the service to terminate (for example, by sending some form of termination signal to it), but does not wait for it to do so. Since the remaining processes of the services are killed according to *KillMode=* and *KillSignal=* or *RestartKillSignal=* as described above immediately after the command exited, this may not result in a clean stop. The specified command should hence be a synchronous operation, not an asynchronous one.

Note that the commands specified in *ExecStop=* are only executed when the service started successfully first. They are not invoked if the service was never started at all, or in case its start-up failed, for example because any of the commands specified in *ExecStart=*, *ExecStartPre=* or *ExecStartPost=* failed (and weren't prefixed with "-", see above) or timed out. Use *ExecStopPost=* to invoke commands when a service failed to start up correctly and is shut down again. Also note that the stop operation is always performed if the service started successfully, even if the processes in the service terminated on their own or were killed. The stop commands must be prepared to deal with that case. *\$MAINPID* will be unset if systemd knows that the main process exited by the time the stop commands are called.

Service restart requests are implemented as stop operations followed by start operations. This means that *ExecStop=* and *ExecStopPost=* are executed during a service restart operation.

It is recommended to use this setting for commands that communicate with the service requesting clean termination. For post-mortem clean-up steps use *ExecStopPost=* instead.

ExecStopPost=

Additional commands that are executed after the service is stopped. This includes cases where the commands configured in *ExecStop=* were used, where the service does not have any *ExecStop=* defined, or where the service exited unexpectedly. This argument takes multiple command lines,

following the same scheme as described for *ExecStart=*. Use of these settings is optional. Specifier and environment variable substitution is supported. Note that – unlike *ExecStop=* – commands specified with this setting are invoked when a service failed to start up correctly and is shut down again.

It is recommended to use this setting for clean-up operations that shall be executed even when the service failed to start up correctly. Commands configured with this setting need to be able to operate even if the service failed starting up half-way and left incompletely initialized data around. As the service's processes have been terminated already when the commands specified with this setting are executed they should not attempt to communicate with them.

Note that all commands that are configured with this setting are invoked with the result code of the service, as well as the main process' exit code and status, set in the *\$SERVICE_RESULT*, *\$EXIT_CODE* and *\$EXIT_STATUS* environment variables, see **systemd.exec(5)** for details.

Note that the execution of *ExecStopPost=* is taken into account for the purpose of *Before=/After=* ordering constraints.

RestartSec=

Configures the time to sleep before restarting a service (as configured with *Restart=*). Takes a unit-less value in seconds, or a time span value such as "5min 20s". Defaults to 100ms.

TimeoutStartSec=

Configures the time to wait for start-up. If a daemon service does not signal start-up completion within the configured time, the service will be considered failed and will be shut down again. The precise action depends on the *TimeoutStartFailureMode=* option. Takes a unit-less value in seconds, or a time span value such as "5min 20s". Pass "infinity" to disable the timeout logic. Defaults to *DefaultTimeoutStartSec=* from the manager configuration file, except when *Type=oneshot* is used, in which case the timeout is disabled by default (see **systemd-system.conf(5)**).

If a service of *Type=notify* sends "EXTEND_TIMEOUT_USEC=...", this may cause the start time to be extended beyond *TimeoutStartSec=*. The first receipt of this message must occur before *TimeoutStartSec=* is exceeded, and once the start time has extended beyond *TimeoutStartSec=*, the service manager will allow the service to continue to start, provided the service repeats "EXTEND_TIMEOUT_USEC=..." within the interval specified until the service startup status is finished by "READY=1". (see **sd_notify(3)**).

TimeoutStopSec=

This option serves two purposes. First, it configures the time to wait for each *ExecStop=* command. If any of them times out, subsequent *ExecStop=* commands are skipped and the service will be terminated by **SIGTERM**. If no *ExecStop=* commands are specified, the service gets the **SIGTERM** immediately. This default behavior can be changed by the *TimeoutStopFailureMode=* option. Second, it configures the time to wait for the service itself to stop. If it doesn't terminate in the specified time, it will be forcibly terminated by **SIGKILL** (see *KillMode=* in **systemd.kill(5)**). Takes a unit-less value in seconds, or a time span value such as "5min 20s". Pass "infinity" to disable the timeout logic. Defaults to *DefaultTimeoutStopSec=* from the manager configuration file (see **systemd-system.conf(5)**).

If a service of *Type=notify* sends "EXTEND_TIMEOUT_USEC=...", this may cause the stop time to be extended beyond *TimeoutStopSec=*. The first receipt of this message must occur before *TimeoutStopSec=* is exceeded, and once the stop time has extended beyond *TimeoutStopSec=*, the service manager will allow the service to continue to stop, provided the service repeats "EXTEND_TIMEOUT_USEC=..." within the interval specified, or terminates itself (see **sd_notify(3)**).

TimeoutAbortSec=

This option configures the time to wait for the service to terminate when it was aborted due to a

watchdog timeout (see *WatchdogSec*=). If the service has a short *TimeoutStopSec*= this option can be used to give the system more time to write a core dump of the service. Upon expiration the service will be forcibly terminated by **SIGKILL** (see *KillMode*= in **systemd.kill(5)**). The core file will be truncated in this case. Use *TimeoutAbortSec*= to set a sensible timeout for the core dumping per service that is large enough to write all expected data while also being short enough to handle the service failure in due time.

Takes a unit-less value in seconds, or a time span value such as "5min 20s". Pass an empty value to skip the dedicated watchdog abort timeout handling and fall back *TimeoutStopSec*=. Pass "infinity" to disable the timeout logic. Defaults to *DefaultTimeoutAbortSec*= from the manager configuration file (see **systemd-system.conf(5)**).

If a service of *Type=notify* handles **SIGABRT** itself (instead of relying on the kernel to write a core dump) it can send "EXTEND_TIMEOUT_USEC=..." to extend the abort time beyond *TimeoutAbortSec*=. The first receipt of this message must occur before *TimeoutAbortSec*= is exceeded, and once the abort time has extended beyond *TimeoutAbortSec*=, the service manager will allow the service to continue to abort, provided the service repeats "EXTEND_TIMEOUT_USEC=..." within the interval specified, or terminates itself (see **sd_notify(3)**).

TimeoutSec=

A shorthand for configuring both *TimeoutStartSec*= and *TimeoutStopSec*= to the specified value.

TimeoutStartFailureMode=, *TimeoutStopFailureMode*=

These options configure the action that is taken in case a daemon service does not signal start-up within its configured *TimeoutStartSec*=, respectively if it does not stop within *TimeoutStopSec*=. Takes one of **terminate**, **abort** and **kill**. Both options default to **terminate**.

If **terminate** is set the service will be gracefully terminated by sending the signal specified in *KillSignal*= (defaults to **SIGTERM**, see **systemd.kill(5)**). If the service does not terminate the *FinalKillSignal*= is sent after *TimeoutStopSec*=. If **abort** is set, *WatchdogSignal*= is sent instead and *TimeoutAbortSec*= applies before sending *FinalKillSignal*=. This setting may be used to analyze services that fail to start-up or shut-down intermittently. By using **kill** the service is immediately terminated by sending *FinalKillSignal*= without any further timeout. This setting can be used to expedite the shutdown of failing services.

RuntimeMaxSec=

Configures a maximum time for the service to run. If this is used and the service has been active for longer than the specified time it is terminated and put into a failure state. Note that this setting does not have any effect on *Type=oneshot* services, as they terminate immediately after activation completed. Pass "infinity" (the default) to configure no runtime limit.

If a service of *Type=notify* sends "EXTEND_TIMEOUT_USEC=...", this may cause the runtime to be extended beyond *RuntimeMaxSec*=. The first receipt of this message must occur before *RuntimeMaxSec*= is exceeded, and once the runtime has extended beyond *RuntimeMaxSec*=, the service manager will allow the service to continue to run, provided the service repeats "EXTEND_TIMEOUT_USEC=..." within the interval specified until the service shutdown is achieved by "STOPPING=1" (or termination). (see **sd_notify(3)**).

WatchdogSec=

Configures the watchdog timeout for a service. The watchdog is activated when the start-up is completed. The service must call **sd_notify(3)** regularly with "WATCHDOG=1" (i.e. the "keep-alive ping"). If the time between two such calls is larger than the configured time, then the service is placed in a failed state and it will be terminated with **SIGABRT** (or the signal specified by *WatchdogSignal*=). By setting *Restart*= to **on-failure**, **on-watchdog**, **on-abnormal** or **always**, the service will be automatically restarted. The time configured here will be passed to the executed service process in the *WATCHDOG_USEC*= environment variable. This allows daemons to automatically enable the keep-alive pinging logic if watchdog support is enabled for the service. If this option is

used, *NotifyAccess*= (see below) should be set to open access to the notification socket provided by systemd. If *NotifyAccess*= is not set, it will be implicitly set to **main**. Defaults to 0, which disables this feature. The service can check whether the service manager expects watchdog keep-alive notifications. See **sd_watchdog_enabled(3)** for details. **sd_event_set_watchdog(3)** may be used to enable automatic watchdog notification support.

Restart=

Configures whether the service shall be restarted when the service process exits, is killed, or a timeout is reached. The service process may be the main service process, but it may also be one of the processes specified with *ExecStartPre*=, *ExecStartPost*=, *ExecStop*=, *ExecStopPost*=, or *ExecReload*=. When the death of the process is a result of systemd operation (e.g. service stop or restart), the service will not be restarted. Timeouts include missing the watchdog "keep-alive ping" deadline and a service start, reload, and stop operation timeouts.

Takes one of **no**, **on-success**, **on-failure**, **on-abnormal**, **on-watchdog**, **on-abort**, or **always**. If set to **no** (the default), the service will not be restarted. If set to **on-success**, it will be restarted only when the service process exits cleanly. In this context, a clean exit means any of the following:

- exit code of 0;
- for types other than *Type=oneshot*, one of the signals **SIGHUP**, **SIGINT**, **SIGTERM**, or **SIGPIPE**;
- exit statuses and signals specified in *SuccessExitStatus*=.

If set to **on-failure**, the service will be restarted when the process exits with a non-zero exit code, is terminated by a signal (including on core dump, but excluding the aforementioned four signals), when an operation (such as service reload) times out, and when the configured watchdog timeout is triggered. If set to **on-abnormal**, the service will be restarted when the process is terminated by a signal (including on core dump, excluding the aforementioned four signals), when an operation times out, or when the watchdog timeout is triggered. If set to **on-abort**, the service will be restarted only if the service process exits due to an uncaught signal not specified as a clean exit status. If set to **on-watchdog**, the service will be restarted only if the watchdog timeout for the service expires. If set to **always**, the service will be restarted regardless of whether it exited cleanly or not, got terminated abnormally by a signal, or hit a timeout.

Table 2. Exit causes and the effect of the *Restart*= settings

Restart settings/Exit causes	no	always	on-success	on-failure	on-abnormal	on-abort	on-watchdog
Clean exit code or signal		X	X				
Unclean exit code		X		X			
Unclean signal		X		X	X	X	
Timeout		X		X	X		
Watchdog		X		X	X		X

As exceptions to the setting above, the service will not be restarted if the exit code or signal is specified in *RestartPreventExitStatus*= (see below) or the service is stopped with **systemctl stop** or an equivalent operation. Also, the services will always be restarted if the exit code or signal is specified in *RestartForceExitStatus*= (see below).

Note that service restart is subject to unit start rate limiting configured with *StartLimitIntervalSec*=

and *StartLimitBurst*=, see **systemd.unit(5)** for details. A restarted service enters the failed state only after the start limits are reached.

Setting this to **on-failure** is the recommended choice for long-running services, in order to increase reliability by attempting automatic recovery from errors. For services that shall be able to terminate on their own choice (and avoid immediate restarting), **on-abnormal** is an alternative choice.

SuccessExitStatus=

Takes a list of exit status definitions that, when returned by the main service process, will be considered successful termination, in addition to the normal successful exit status 0 and, except for *Type=oneshot*, the signals **SIGHUP**, **SIGINT**, **SIGTERM**, and **SIGPIPE**. Exit status definitions can be numeric termination statuses, termination status names, or termination signal names, separated by spaces. See the Process Exit Codes section in **systemd.exec(5)** for a list of termination status names (for this setting only the part without the "EXIT_" or "EX_" prefix should be used). See **signal(7)** for a list of signal names.

Note that this setting does not change the mapping between numeric exit statuses and their names, i.e. regardless how this setting is used 0 will still be mapped to "SUCCESS" (and thus typically shown as "0/SUCCESS" in tool outputs) and 1 to "FAILURE" (and thus typically shown as "1/FAILURE"), and so on. It only controls what happens as effect of these exit statuses, and how it propagates to the state of the service as a whole.

This option may appear more than once, in which case the list of successful exit statuses is merged. If the empty string is assigned to this option, the list is reset, all prior assignments of this option will have no effect.

Example 1. A service with the *SuccessExitStatus*= setting

```
SuccessExitStatus=TEMPFAIL 250 SIGKILL
```

Exit status 75 (**TEMPFAIL**), 250, and the termination signal **SIGKILL** are considered clean service terminations.

Note: **systemd-analyze exit-status** may be used to list exit statuses and translate between numerical status values and names.

RestartPreventExitStatus=

Takes a list of exit status definitions that, when returned by the main service process, will prevent automatic service restarts, regardless of the restart setting configured with *Restart*=. Exit status definitions can either be numeric exit codes or termination signal names, and are separated by spaces. Defaults to the empty list, so that, by default, no exit status is excluded from the configured restart logic. For example:

```
RestartPreventExitStatus=1 6 SIGABRT
```

ensures that exit codes 1 and 6 and the termination signal **SIGABRT** will not result in automatic service restarting. This option may appear more than once, in which case the list of restart-preventing statuses is merged. If the empty string is assigned to this option, the list is reset and all prior assignments of this option will have no effect.

Note that this setting has no effect on processes configured via *ExecStartPre*=, *ExecStartPost*=, *ExecStop*=, *ExecStopPost*= or *ExecReload*=, but only on the main service process, i.e. either the one invoked by *ExecStart*= or (depending on *Type*=, *PIDFile*=, ...) the otherwise configured main process.

RestartForceExitStatus=

Takes a list of exit status definitions that, when returned by the main service process, will force automatic service restarts, regardless of the restart setting configured with *Restart*=. The argument

format is similar to *RestartPreventExitStatus*=.

RootDirectoryStartOnly=

Takes a boolean argument. If true, the root directory, as configured with the *RootDirectory*= option (see **systemd.exec(5)** for more information), is only applied to the process started with *ExecStart*=, and not to the various other *ExecStartPre*=, *ExecStartPost*=, *ExecReload*=, *ExecStop*=, and *ExecStopPost*= commands. If false, the setting is applied to all configured commands the same way. Defaults to false.

NonBlocking=

Set the **O_NONBLOCK** flag for all file descriptors passed via socket-based activation. If true, all file descriptors ≥ 3 (i.e. all except stdin, stdout, stderr), excluding those passed in via the file descriptor storage logic (see *FileDescriptorStoreMax*= for details), will have the **O_NONBLOCK** flag set and hence are in non-blocking mode. This option is only useful in conjunction with a socket unit, as described in **systemd.socket(5)** and has no effect on file descriptors which were previously saved in the file-descriptor store for example. Defaults to false.

NotifyAccess=

Controls access to the service status notification socket, as accessible via the **sd_notify**(3) call. Takes one of **none** (the default), **main**, **exec** or **all**. If **none**, no daemon status updates are accepted from the service processes, all status update messages are ignored. If **main**, only service updates sent from the main process of the service are accepted. If **exec**, only service updates sent from any of the main or control processes originating from one of the *Exec*=* commands are accepted. If **all**, all services updates from all members of the service's control group are accepted. This option should be set to open access to the notification socket when using *Type=notify* or *WatchdogSec*= (see above). If those options are used but *NotifyAccess*= is not configured, it will be implicitly set to **main**.

Note that **sd_notify()** notifications may be attributed to units correctly only if either the sending process is still around at the time PID 1 processes the message, or if the sending process is explicitly runtime-tracked by the service manager. The latter is the case if the service manager originally forked off the process, i.e. on all processes that match **main** or **exec**. Conversely, if an auxiliary process of the unit sends an **sd_notify()** message and immediately exits, the service manager might not be able to properly attribute the message to the unit, and thus will ignore it, even if *NotifyAccess=all* is set for it.

Hence, to eliminate all race conditions involving lookup of the client's unit and attribution of notifications to units correctly, **sd_notify_barrier()** may be used. This call acts as a synchronization point and ensures all notifications sent before this call have been picked up by the service manager when it returns successfully. Use of **sd_notify_barrier()** is needed for clients which are not invoked by the service manager, otherwise this synchronization mechanism is unnecessary for attribution of notifications to the unit.

Sockets=

Specifies the name of the socket units this service shall inherit socket file descriptors from when the service is started. Normally, it should not be necessary to use this setting, as all socket file descriptors whose unit shares the same name as the service (subject to the different unit name suffix of course) are passed to the spawned process.

Note that the same socket file descriptors may be passed to multiple processes simultaneously. Also note that a different service may be activated on incoming socket traffic than the one which is ultimately configured to inherit the socket file descriptors. Or, in other words: the *Service*= setting of .socket units does not have to match the inverse of the *Sockets*= setting of the .service it refers to.

This option may appear more than once, in which case the list of socket units is merged. Note that once set, clearing the list of sockets again (for example, by assigning the empty string to this option) is not supported.

FileDescriptorStoreMax=

Configure how many file descriptors may be stored in the service manager for the service using `sd_pid_notify_with_fds(3)`'s "FDSTORE=1" messages. This is useful for implementing services that can restart after an explicit request or a crash without losing state. Any open sockets and other file descriptors which should not be closed during the restart may be stored this way. Application state can either be serialized to a file in /run/, or better, stored in a `memfd_create(2)` memory file descriptor. Defaults to 0, i.e. no file descriptors may be stored in the service manager. All file descriptors passed to the service manager from a specific service are passed back to the service's main process on the next service restart (see `sd_listen_fds(3)` for details about the precise protocol used and the order in which the file descriptors are passed). Any file descriptors passed to the service manager are automatically closed when **POLLHUP** or **POLLERR** is seen on them, or when the service is fully stopped and no job is queued or being executed for it. If this option is used, `NotifyAccess=` (see above) should be set to open access to the notification socket provided by systemd. If `NotifyAccess=` is not set, it will be implicitly set to **main**.

USBFunctionDescriptors=

Configure the location of a file containing **USB FunctionFS**^[2] descriptors, for implementation of USB gadget functions. This is used only in conjunction with a socket unit with `ListenUSBFunction=` configured. The contents of this file are written to the ep0 file after it is opened.

USBFunctionStrings=

Configure the location of a file containing USB FunctionFS strings. Behavior is similar to `USBFunctionDescriptors=` above.

OOMPolicy=

Configure the Out-Of-Memory (OOM) killer policy. On Linux, when memory becomes scarce the kernel might decide to kill a running process in order to free up memory and reduce memory pressure. This setting takes one of **continue**, **stop** or **kill**. If set to **continue** and a process of the service is killed by the kernel's OOM killer this is logged but the service continues running. If set to **stop** the event is logged but the service is terminated cleanly by the service manager. If set to **kill** and one of the service's processes is killed by the OOM killer the kernel is instructed to kill all remaining processes of the service, too. Defaults to the setting `DefaultOOMPolicy=` in `systemd-system.conf(5)` is set to, except for services where `Delegate=` is turned on, where it defaults to **continue**.

Use the `OOMScoreAdjust=` setting to configure whether processes of the unit shall be considered preferred or less preferred candidates for process termination by the Linux OOM killer logic. See `systemd.exec(5)` for details.

Check `systemd.exec(5)` and `systemd.kill(5)` for more settings.

COMMAND LINES

This section describes command line parsing and variable and specifier substitutions for `ExecStart=`, `ExecStartPre=`, `ExecStartPost=`, `ExecReload=`, `ExecStop=`, and `ExecStopPost=` options.

Multiple command lines may be concatenated in a single directive by separating them with semicolons (these semicolons must be passed as separate words). Lone semicolons may be escaped as "\;".

Each command line is unquoted using the rules described in "Quoting" section in `systemd.syntax(7)`. The first item becomes the command to execute, and the subsequent items the arguments.

This syntax is inspired by shell syntax, but only the meta-characters and expansions described in the following paragraphs are understood, and the expansion of variables is different. Specifically, redirection using "<", "<<", ">", and ">>", pipes using "|", running programs in the background using "&", and *other elements of shell syntax are not supported*.

The command to execute may contain spaces, but control characters are not allowed.

The command line accepts "%" specifiers as described in `systemd.unit(5)`.

Basic environment variable substitution is supported. Use "\${FOO}" as part of a word, or as a word of its own, on the command line, in which case it will be erased and replaced by the exact value of the environment variable (if any) including all whitespace it contains, always resulting in exactly a single

argument. Use "\$FOO" as a separate word on the command line, in which case it will be replaced by the value of the environment variable split at whitespace, resulting in zero or more arguments. For this type of expansion, quotes are respected when splitting into words, and afterwards removed.

If the command is not a full (absolute) path, it will be resolved to a full path using a fixed search path determined at compilation time. Searched directories include /usr/local/bin/, /usr/bin/, /bin/ on systems using split /usr/bin/ and /bin/ directories, and their sbin/ counterparts on systems using split bin/ and sbin/. It is thus safe to use just the executable name in case of executables located in any of the "standard" directories, and an absolute path must be used in other cases. Using an absolute path is recommended to avoid ambiguity. Hint: this search path may be queried using **systemd-path search-binaries-default**.

Example:

```
Environment="ONE=one" "TWO=two two"
ExecStart=echo $ONE ${TWO}
```

This will execute **/bin/echo** with four arguments: "one", "two", "two", and "two two".

Example:

```
Environment=ONE='one' "TWO='two two' too" THREE=
ExecStart=/bin/echo ${ONE} ${TWO} ${THREE}
ExecStart=/bin/echo $ONE ${TWO} ${THREE}
```

This results in **/bin/echo** being called twice, the first time with arguments "'one'", "'two two' too", "", and the second time with arguments "one", "two two", "too".

To pass a literal dollar sign, use "\$\$". Variables whose value is not known at expansion time are treated as empty strings. Note that the first argument (i.e. the program to execute) may not be a variable.

Variables to be used in this fashion may be defined through *Environment=* and *EnvironmentFile=*. In addition, variables listed in the section "Environment variables in spawned processes" in **systemd.exec(5)**, which are considered "static configuration", may be used (this includes e.g. *\$USER*, but not *\$TERM*).

Note that shell command lines are not directly supported. If shell command lines are to be used, they need to be passed explicitly to a shell implementation of some kind. Example:

```
ExecStart=sh -c 'dmesg | tac'
```

Example:

```
ExecStart=echo one ; echo "two two"
```

This will execute **echo** two times, each time with one argument: "one" and "two two", respectively. Because two commands are specified, *Type=oneshot* must be used.

Example:

```
ExecStart=echo / >/dev/null & \
ls
```

This will execute **echo** with five arguments: "/", ">/dev/null", "&", ";", and "ls".

EXAMPLES

Example 2. Simple service

The following unit file creates a service that will execute **/usr/sbin/foo-daemon**. Since no *Type=* is specified, the default *Type=simple* will be assumed. **systemd** will assume the unit to be started immediately after the program has begun executing.

```
[Unit]
Description=Foo
```

```
[Service]
ExecStart=/usr/sbin/foo-daemon
```

```
[Install]
WantedBy=multi-user.target
```

Note that systemd assumes here that the process started by systemd will continue running until the service terminates. If the program daemonizes itself (i.e. forks), please use *Type=forking* instead.

Since no *ExecStop=* was specified, systemd will send SIGTERM to all processes started from this service, and after a timeout also SIGKILL. This behavior can be modified, see **systemd.kill(5)** for details.

Note that this unit type does not include any type of notification when a service has completed initialization. For this, you should use other unit types, such as *Type=notify* if the service understands systemd's notification protocol, *Type=forking* if the service can background itself or *Type=dbus* if the unit acquires a DBus name once initialization is complete. See below.

Example 3. Oneshot service

Sometimes, units should just execute an action without keeping active processes, such as a filesystem check or a cleanup action on boot. For this, *Type=oneshot* exists. Units of this type will wait until the process specified terminates and then fall back to being inactive. The following unit will perform a cleanup action:

```
[Unit]
Description=Cleanup old Foo data
```

```
[Service]
Type=oneshot
ExecStart=/usr/sbin/foo-cleanup
```

```
[Install]
WantedBy=multi-user.target
```

Note that systemd will consider the unit to be in the state "starting" until the program has terminated, so ordered dependencies will wait for the program to finish before starting themselves. The unit will revert to the "inactive" state after the execution is done, never reaching the "active" state. That means another request to start the unit will perform the action again.

Type=oneshot are the only service units that may have more than one *ExecStart=* specified. For units with multiple commands (*Type=oneshot*), all commands will be run again.

For *Type=oneshot*, *Restart=always* and *Restart=on-success* are *not* allowed.

Example 4. Stoppable oneshot service

Similarly to the oneshot services, there are sometimes units that need to execute a program to set up something and then execute another to shut it down, but no process remains active while they are considered "started". Network configuration can sometimes fall into this category. Another use case is if a oneshot service shall not be executed each time when they are pulled in as a dependency, but only the first time.

For this, systemd knows the setting *RemainAfterExit=yes*, which causes systemd to consider the unit to be active if the start action exited successfully. This directive can be used with all types, but is most useful with *Type=oneshot* and *Type=simple*. With *Type=oneshot*, systemd waits until the start action has completed before it considers the unit to be active, so dependencies start only after the start action has succeeded. With *Type=simple*, dependencies will start immediately after the start action has been dispatched. The following unit provides an example for a simple static firewall.

```
[Unit]
Description=Simple firewall
```

```
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/sbin/simple-firewall-start
ExecStop=/usr/local/sbin/simple-firewall-stop
```

```
[Install]
WantedBy=multi-user.target
```

Since the unit is considered to be running after the start action has exited, invoking **systemctl start** on that unit again will cause no action to be taken.

Example 5. Traditional forking services

Many traditional daemons/services background (i.e. fork, daemonize) themselves when starting. Set **Type=forking** in the service's unit file to support this mode of operation. systemd will consider the service to be in the process of initialization while the original program is still running. Once it exits successfully and at least a process remains (and *RemainAfterExit=no*), the service is considered started.

Often, a traditional daemon only consists of one process. Therefore, if only one process is left after the original process terminates, systemd will consider that process the main process of the service. In that case, the **\$MAINPID** variable will be available in *ExecReload*=, *ExecStop*=, etc.

In case more than one process remains, systemd will be unable to determine the main process, so it will not assume there is one. In that case, **\$MAINPID** will not expand to anything. However, if the process decides to write a traditional PID file, systemd will be able to read the main PID from there. Please set **PIDFile**= accordingly. Note that the daemon should write that file before finishing with its initialization. Otherwise, systemd might try to read the file before it exists.

The following example shows a simple daemon that forks and just starts one process in the background:

```
[Unit]
Description=Some simple daemon
```

```
[Service]
Type=forking
ExecStart=/usr/sbin/my-simple-daemon -d
```

```
[Install]
WantedBy=multi-user.target
```

Please see **systemd.kill(5)** for details on how you can influence the way systemd terminates the service.

Example 6. DBus services

For services that acquire a name on the DBus system bus, use **Type=dbus** and set **BusName**= accordingly. The service should not fork (daemonize). systemd will consider the service to be initialized once the name has been acquired on the system bus. The following example shows a typical DBus service:

```
[Unit]
Description=Simple DBus service
```

```
[Service]
Type=dbus
BusName=org.example.simple-dbus-service
ExecStart=/usr/sbin/simple-dbus-service
```

```
[Install]
WantedBy=multi-user.target
```

For *bus-activatable* services, do not include a [Install] section in the systemd service file, but use the *SystemdService=* option in the corresponding DBus service file, for example (`/usr/share/dbus-1/system-services/org.example.simple-dbus-service.service`):

```
[D-BUS Service]
Name=org.example.simple-dbus-service
Exec=/usr/sbin/simple-dbus-service
User=root
SystemdService=simple-dbus-service.service
```

Please see **systemd.kill(5)** for details on how you can influence the way systemd terminates the service.

Example 7. Services that notify systemd about their initialization

Type=simple services are really easy to write, but have the major disadvantage of systemd not being able to tell when initialization of the given service is complete. For this reason, systemd supports a simple notification protocol that allows daemons to make systemd aware that they are done initializing. Use *Type=notify* for this. A typical service file for such a daemon would look like this:

```
[Unit]
Description=Simple notifying service

[Service]
Type=notify
ExecStart=/usr/sbin/simple-notifying-service

[Install]
WantedBy=multi-user.target
```

Note that the daemon has to support systemd's notification protocol, else systemd will think the service has not started yet and kill it after a timeout. For an example of how to update daemons to support this protocol transparently, take a look at **sd_notify(3)**. systemd will consider the unit to be in the 'starting' state until a readiness notification has arrived.

Please see **systemd.kill(5)** for details on how you can influence the way systemd terminates the service.

SEE ALSO

systemd(1), systemctl(1), systemd-system.conf(5), systemd.unit(5), systemd.exec(5), systemd.resource-control(5), systemd.kill(5), systemd.directives(7), systemd-run(1)

NOTES

1. Incompatibilities with SysV
<https://www.freedesktop.org/wiki/Software/systemd/Incompatibilities>
2. USB FunctionFS
<https://www.kernel.org/doc/Documentation/usb/functions.txt>

NAME

systemd.time – Time and date specifications

DESCRIPTION

In systemd, timestamps, time spans, and calendar events are displayed and may be specified in closely related syntaxes.

DISPLAYING TIME SPANS

Time spans refer to time durations. On display, systemd will present time spans as a space-separated series of time values each suffixed by a time unit. Example:

2h 30min

All specified time values are meant to be added up. The above hence refers to 150 minutes. Display is locale-independent, only English names for the time units are used.

PARSING TIME SPANS

When parsing, systemd will accept the same time span syntax. Separating spaces may be omitted. The following time units are understood:

- usec, us, μ s
- msec, ms
- seconds, second, sec, s
- minutes, minute, min, m
- hours, hour, hr, h
- days, day, d
- weeks, week, w
- months, month, M (defined as 30.44 days)
- years, year, y (defined as 365.25 days)

If no time unit is specified, generally seconds are assumed, but some exceptions exist and are marked as such. In a few cases "ns", "nsec" is accepted too, where the granularity of the time span permits this. Parsing is generally locale-independent, non-English names for the time units are not accepted.

Examples for valid time span specifications:

2 h

2hours

48hr

1y 12month

55s500ms

300ms20s 5day

One can use the **timespan** command of **systemd-analyze(1)** to normalise a textual time span for testing and validation purposes.

Internally, systemd generally operates with microsecond time granularity, while the default time unit in user-configurable time spans is usually seconds (see above). This disparity becomes visible when comparing the same settings in the (high-level) unit file syntax with the matching (more low-level) D-Bus properties (which are what **systemctl(1)**'s **show** command displays). The former typically are suffixed with "...Sec" to indicate the default unit of seconds, the latter are typically suffixed with "...USec" to indicate the underlying low-level time unit, even if they both encapsulate the very same settings.

DISPLAYING TIMESTAMPSP

Timestamps refer to specific, unique points in time. On display, systemd will format these in the local timezone as follows:

Fri 2012-11-23 23:02:15 CET

The weekday is printed in the abbreviated English language form. The formatting is locale-independent.

In some cases timestamps are shown in the UTC timezone instead of the local timezone, which is indicated via the "UTC" timezone specifier in the output.

In some cases timestamps are shown with microsecond granularity. In this case the sub-second remainder is separated by a full stop from the seconds component.

PARSING TIMESTAMPS

When parsing, systemd will accept a similar syntax, but expects no timezone specification, unless it is given as the literal string "UTC" (for the UTC timezone), or is specified to be the locally configured timezone, or the timezone name in the IANA timezone database format. The complete list of timezones supported on your system can be obtained using the "timedatectl list-timezones" (see [timedatectl\(1\)](#)). Using IANA format is recommended over local timezone names, as less prone to errors (e.g. with local timezone it's possible to specify daylight saving time in winter, even though that is not correct). The weekday specification is optional, but when the weekday is specified, it must either be in the abbreviated ("Wed") or non-abbreviated ("Wednesday") English language form (case does not matter), and is not subject to the locale choice of the user. Either the date, or the time part may be omitted, in which case the current date or 00:00:00, respectively, is assumed. The seconds component of the time may also be omitted, in which case ":00" is assumed. Year numbers may be specified in full or may be abbreviated (omitting the century).

A timestamp is considered invalid if a weekday is specified and the date does not match the specified day of the week.

When parsing, systemd will also accept a few special placeholders instead of timestamps: "now" may be used to refer to the current time (or of the invocation of the command that is currently executed). "today", "yesterday", and "tomorrow" refer to 00:00:00 of the current day, the day before, or the next day, respectively.

When parsing, systemd will also accept relative time specifications. A time span (see above) that is prefixed with "+" is evaluated to the current time plus the specified time span. Correspondingly, a time span that is prefixed with "-" is evaluated to the current time minus the specified time span. Instead of prefixing the time span with "+" or "-", it may also be suffixed with a space and the word "left" or "ago".

Finally, a timespan prefixed with "@" is evaluated relative to the UNIX time epoch 1st Jan, 1970, 00:00.

Examples for valid timestamps and their normalized form (assuming the current time was 2012-11-23 18:15:22 and the timezone was UTC+8, for example "TZ=:Asia/Shanghai"):

```

Fri 2012-11-23 11:12:13 → Fri 2012-11-23 11:12:13
2012-11-23 11:12:13 → Fri 2012-11-23 11:12:13
2012-11-23 11:12:13 UTC → Fri 2012-11-23 19:12:13
2012-11-23 → Fri 2012-11-23 00:00:00
12-11-23 → Fri 2012-11-23 00:00:00
11:12:13 → Fri 2012-11-23 11:12:13
11:12 → Fri 2012-11-23 11:12:00
now → Fri 2012-11-23 18:15:22
today → Fri 2012-11-23 00:00:00
today UTC → Fri 2012-11-23 16:00:00
yesterday → Fri 2012-11-22 00:00:00
tomorrow → Fri 2012-11-24 00:00:00
tomorrow Pacific/Auckland → Thu 2012-11-23 19:00:00
+3h30min → Fri 2012-11-23 21:45:22
-5s → Fri 2012-11-23 18:15:17
11min ago → Fri 2012-11-23 18:04:22
@1395716396 → Tue 2014-03-25 03:59:56

```

Note that timestamps displayed by remote systems with a non-matching timezone are usually not parsable

locally, as the timezone component is not understood (unless it happens to be "UTC").

Timestamps may also be specified with microsecond granularity. The sub-second remainder is expected separated by a full stop from the seconds component. Example:

2014-03-25 03:59:56.654563

In some cases, systemd will display a relative timestamp (relative to the current time, or the time of invocation of the command) instead of or in addition to an absolute timestamp as described above. A relative timestamp is formatted as follows:

2 months 5 days ago

Note that a relative timestamp is also accepted where a timestamp is expected (see above).

Use the **timestamp** command of **systemd-analyze(1)** to validate and normalize timestamps for testing purposes.

CALENDAR EVENTS

Calendar events may be used to refer to one or more points in time in a single expression. They form a superset of the absolute timestamps explained above:

Thu,Fri 2012-*–1,5 11:12:13

The above refers to 11:12:13 of the first or fifth day of any month of the year 2012, but only if that day is a Thursday or Friday.

The weekday specification is optional. If specified, it should consist of one or more English language weekday names, either in the abbreviated (Wed) or non-abbreviated (Wednesday) form (case does not matter), separated by commas. Specifying two weekdays separated by ".." refers to a range of continuous weekdays. "," and ".." may be combined freely.

In the date and time specifications, any component may be specified as "*" in which case any value will match. Alternatively, each component can be specified as a list of values separated by commas. Values may be suffixed with "/" and a repetition value, which indicates that the value itself and the value plus all multiples of the repetition value are matched. Two values separated by ".." may be used to indicate a range of values; ranges may also be followed with "/" and a repetition value, in which case the expression matches all times starting with the start value, and continuing with all multiples of the repetition value relative to the start value, ending at the end value the latest.

A date specification may use "~" to indicate the last day(s) in a month. For example, "*–02~03" means "the third last day in February," and "Mon *–05~07/1" means "the last Monday in May."

The seconds component may contain decimal fractions both in the value and the repetition. All fractions are rounded to 6 decimal places.

Either time or date specification may be omitted, in which case the current day and 00:00:00 is implied, respectively. If the second component is not specified, ":00" is assumed.

Timezone can be specified as the literal string "UTC", or the local timezone, similar to the supported syntax of timestamps (see above), or the timezone in the IANA timezone database format (also see above).

The following special expressions may be used as shorthands for longer normalized forms:

```

minutely → *-*-* *.*:00
hourly → *-*-* *:00:00
daily → *-*-* 00:00:00
monthly → *-*–01 00:00:00
weekly → Mon *-*-* 00:00:00
yearly → *–01–01 00:00:00
quarterly → *–01,04,07,10–01 00:00:00
semianually → *–01,07–01 00:00:00

```

Examples for valid timestamps and their normalized form:

```

Sat,Thu,Mon..Wed,Sat..Sun → Mon..Thu,Sat,Sun *-*-* 00:00:00
Mon,Sun 12-*-* 2,1:23 → Mon,Sun 2012-*-* 01,02:23:00
    Wed *-1 → Wed *-*01 00:00:00
    Wed..Wed,Wed *-1 → Wed *-*01 00:00:00
    Wed, 17:48 → Wed *-*-* 17:48:00
Wed..Sat,Tue 12-10-15 1:2:3 → Tue..Sat 2012-10-15 01:02:03
    *-*7 0:0:0 → *-*07 00:00:00
    10-15 → *-10-15 00:00:00
    monday *-12-* 17:00 → Mon *-12-* 17:00:00
Mon,Fri *-*3,1,2 *:30:45 → Mon,Fri *-*01,02,03 *:30:45
    12,14,13,12:20,10,30 → *-*12,13,14:10,20,30:00
    12..14:10,20,30 → *-*12..14:10,20,30:00
mon,fri *-1/2-1,3 *:30:45 → Mon,Fri *-01/2-01,03 *:30:45
    03-05 08:05:40 → *-03-05 08:05:40
    08:05:40 → *-*-* 08:05:40
    05:40 → *-*-* 05:40:00
Sat,Sun 12-05 08:05:40 → Sat,Sun *-12-05 08:05:40
    Sat,Sun 08:05:40 → Sat,Sun *-*-* 08:05:40
    2003-03-05 05:40 → 2003-03-05 05:40:00
05:40:23.420000/3.1700005 → *-*-* 05:40:23.420000/3.170001
    2003-02..04-05 → 2003-02..04-05 00:00:00
2003-03-05 05:40 UTC → 2003-03-05 05:40:00 UTC
    2003-03-05 → 2003-03-05 00:00:00
    03-05 → *-03-05 00:00:00
    hourly → *-*-* *:00:00
    daily → *-*-* 00:00:00
    daily UTC → *-*-* 00:00:00 UTC
    monthly → *-*-*01 00:00:00
    weekly → Mon *-*-* 00:00:00
weekly Pacific/Auckland → Mon *-*-* 00:00:00 Pacific/Auckland
    yearly → *-01-01 00:00:00
    annually → *-01-01 00:00:00
    *:2/3 → *-*-* *:02/3:00

```

Calendar events are used by timer units, see **systemd.timer(5)** for details.

Use the **calendar** command of **systemd-analyze(1)** to validate and normalize calendar time specifications for testing purposes. The tool also calculates when a specified calendar event would occur next.

SEE ALSO

systemd(1), **journalctl(1)**, **systemd.timer(5)**, **systemd.unit(5)**, **systemd.directives(7)**, **systemd-analyze(1)**

NAME

tail – output the last part of files

SYNOPSIS

tail [*OPTION*]... [*FILE*]...

DESCRIPTION

Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is **-**, read standard input.

Mandatory arguments to long options are mandatory for short options too.

-c, --bytes=[+]NUM

output the last NUM bytes; or use **-c +NUM** to output starting with byte NUM of each file

-f, --follow[={name|descriptor}]

output appended data as the file grows;

an absent option argument means 'descriptor'

-F same as **--follow=name --retry**

-n, --lines=[+]NUM

output the last NUM lines, instead of the last 10; or use **-n +NUM** to output starting with line NUM

--max-unchanged-stats=N

with **--follow=name**, reopen a FILE which has not

changed size after N (default 5) iterations to see if it has been unlinked or renamed (this is the usual case of rotated log files); with inotify, this option is rarely useful

--pid=PID

with **-f**, terminate after process ID, PID dies

-q, --quiet, --silent

never output headers giving file names

--retry

keep trying to open a file if it is inaccessible

-s, --sleep-interval=N

with **-f**, sleep for approximately N seconds (default 1.0) between iterations; with inotify and **--pid=P**, check process P at least once every N seconds

-v, --verbose

always output headers giving file names

-z, --zero-terminated

line delimiter is NUL, not newline

--help display this help and exit

--version

output version information and exit

NUM may have a multiplier suffix: b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000, G 1024*1024*1024, and so on for T, P, E, Z, Y. Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

With **--follow** (**-f**), tail defaults to following the file descriptor, which means that even if a tail'ed file is renamed, tail will continue to track its end. This default behavior is not desirable when you really want to track the actual name of the file, not the file descriptor (e.g., log rotation). Use **--follow=name** in that case. That causes tail to track the named file in a way that accommodates renaming, removal and creation.

AUTHOR

Written by Paul Rubin, David MacKenzie, Ian Lance Taylor, and Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

`head(1)`

Full documentation <<https://www.gnu.org/software/coreutils/tail>>
or available locally via: `info '(coreutils) tail invocation'`

NAME

tar – an archiving utility

SYNOPSIS**Traditional usage**

```
tar {A|c|d|r|t|u|x}[GnSkUWOmpsMBiajJzZhPlRvwo] [ARG...]
```

UNIX-style usage

```
tar -A [OPTIONS] ARCHIVE ARCHIVE
```

```
tar -c [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar -d [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar -t [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

```
tar -r [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar -u [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

GNU-style usage

```
tar {--catenate|--concatenate} [OPTIONS] ARCHIVE ARCHIVE
```

```
tar --create [--file ARCHIVE] [OPTIONS] [FILE...]
```

```
tar {--diff|--compare} [--file ARCHIVE] [OPTIONS] [FILE...]
```

```
tar --delete [--file ARCHIVE] [OPTIONS] [MEMBER...]
```

```
tar --append [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar --list [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

```
tar --test-label [--file ARCHIVE] [OPTIONS] [LABEL...]
```

```
tar --update [--file ARCHIVE] [OPTIONS] [FILE...]
```

```
tar --update [-f ARCHIVE] [OPTIONS] [FILE...]
```

```
tar {--extract|--get} [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

NOTE

This manpage is a short description of GNU **tar**. For a detailed discussion, including examples and usage recommendations, refer to the **GNU Tar Manual** available in texinfo format. If the **info** reader and the tar documentation are properly installed on your system, the command

```
info tar
```

should give you access to the complete manual.

You can also view the manual using the info mode in **emacs(1)**, or find it in various formats online at

```
http://www.gnu.org/software/tar/manual
```

If any discrepancies occur between this manpage and the **GNU Tar Manual**, the later shall be considered the authoritative source.

DESCRIPTION

GNU **tar** is an archiving program designed to store multiple files in a single file (an **archive**), and to manipulate such archives. The archive can be either a regular file or a device (e.g. a tape drive, hence the name of the program, which stands for **tape archiver**), which can be located either on the local or on a remote machine.

Option styles

Options to GNU **tar** can be given in three different styles. In **traditional style**, the first argument is a cluster of option letters and all subsequent arguments supply arguments to those options that require them. The arguments are read in the same order as the option letters. Any command line words that remain after all options have been processed are treated as non-optional arguments: file or archive member names.

For example, the **c** option requires creating the archive, the **v** option requests the verbose operation, and the **f** option takes an argument that sets the name of the archive to operate upon. The following command, written in the traditional style, instructs tar to store all files from the directory **/etc** into the archive file **etc.tar** verbosely listing the files being archived:

```
tar cfv etc.tar /etc
```

In **UNIX** or **short-option style**, each option letter is prefixed with a single dash, as in other command line utilities. If an option takes argument, the argument follows it, either as a separate command line word, or immediately following the option. However, if the option takes an **optional** argument, the argument must follow the option letter without any intervening whitespace, as in **-g/tmp/snar.db**.

Any number of options not taking arguments can be clustered together after a single dash, e.g. **-vkp**. Options that take arguments (whether mandatory or optional), can appear at the end of such a cluster, e.g. **-vkpf a.tar**.

The example command above written in the **short-option style** could look like:

```
tar -cvf etc.tar /etc
or
tar -c -v -f etc.tar /etc
```

In **GNU** or **long-option style**, each option begins with two dashes and has a meaningful name, consisting of lower-case letters and dashes. When used, the long option can be abbreviated to its initial letters, provided that this does not create ambiguity. Arguments to long options are supplied either as a separate command line word, immediately following the option, or separated from the option by an equals sign with no intervening whitespace. Optional arguments must always use the latter method.

Here are several ways of writing the example command in this style:

```
tar --create --file etc.tar --verbose /etc
or (abbreviating some options):
tar --cre --file=etc.tar --verb /etc
```

The options in all three styles can be intermixed, although doing so with old options is not encouraged.

Operation mode

The options listed in the table below tell GNU **tar** what operation it is to perform. Exactly one of them must be given. Meaning of non-optional arguments depends on the operation mode requested.

-A, --catenate, --concatenate

Append archive to the end of another archive. The arguments are treated as the names of archives to append. All archives must be of the same format as the archive they are appended to, otherwise the resulting archive might be unusable with non-GNU implementations of **tar**. Notice also that when more than one archive is given, the members from archives other than the first one will be accessible in the resulting archive only if using the **-i** (**--ignore-zeros**) option.

Compressed archives cannot be concatenated.

-c, --create

Create a new archive. Arguments supply the names of the files to be archived. Directories are archived recursively, unless the **--no-recursion** option is given.

-d, --diff, --compare

Find differences between archive and file system. The arguments are optional and specify archive members to compare. If not given, the current working directory is assumed.

--delete

Delete from the archive. The arguments supply names of the archive members to be removed. At least one argument must be given.

This option does not operate on compressed archives. There is no short option equivalent.

-r, --append

Append files to the end of an archive. Arguments have the same meaning as for **-c** (**--create**).

-t, --list

List the contents of an archive. Arguments are optional. When given, they specify the names of the members to list.

--test-label

Test the archive volume label and exit. When used without arguments, it prints the volume label (if any) and exits with status **0**. When one or more command line arguments are given, **tar** compares the volume label with each argument. It exits with code **0** if a match is found, and with code **1** otherwise. No output is displayed, unless used together with the **-v** (**--verbose**) option.

There is no short option equivalent for this option.

-u, --update

Append files which are newer than the corresponding copy in the archive. Arguments have the same meaning as with **-c** and **-r** options. Notice, that newer files don't replace their old archive copies, but instead are appended to the end of archive. The resulting archive can thus contain several members of the same name, corresponding to various versions of the same file.

-x, --extract, --get

Extract files from an archive. Arguments are optional. When given, they specify names of the archive members to be extracted.

--show-defaults

Show built-in defaults for various **tar** options and exit. No arguments are allowed.

-?, --help

Display a short option summary and exit. No arguments allowed.

--usage

Display a list of available options and exit. No arguments allowed.

--version

Print program version and copyright information and exit.

OPTIONS**Operation modifiers****--check-device**

Check device numbers when creating incremental archives (default).

-g, --listed-incremental=FILE

Handle new GNU-format incremental backups. *FILE* is the name of a **snapshot file**, where tar stores additional information which is used to decide which files changed since the previous incremental dump and, consequently, must be dumped again. If *FILE* does not exist when creating an archive, it will be created and all files will be added to the resulting archive (the **level 0** dump). To create incremental archives of non-zero level *N*, create a copy of the snapshot file created during

the level **N-1**, and use it as *FILE*.

When listing or extracting, the actual contents of *FILE* is not inspected, it is needed only due to syntactical requirements. It is therefore common practice to use **/dev/null** in its place.

--hole-detection=METHOD

Use *METHOD* to detect holes in sparse files. This option implies **--sparse**. Valid values for *METHOD* are **seek** and **raw**. Default is **seek** with fallback to **raw** when not applicable.

-G, --incremental

Handle old GNU-format incremental backups.

--ignore-failed-read

Do not exit with nonzero on unreadable files.

--level=NUMBER

Set dump level for created listed-incremental archive. Currently only **--level=0** is meaningful: it instructs **tar** to truncate the snapshot file before dumping, thereby forcing a level 0 dump.

-n, --seek

Assume the archive is seekable. Normally **tar** determines automatically whether the archive can be seeked or not. This option is intended for use in cases when such recognition fails. It takes effect only if the archive is open for reading (e.g. with **--list** or **--extract** options).

--no-check-device

Do not check device numbers when creating incremental archives.

--no-seek

Assume the archive is not seekable.

--occurrence[=N]

Process only the *N*th occurrence of each file in the archive. This option is valid only when used with one of the following subcommands: **--delete**, **--diff**, **--extract** or **--list** and when a list of files is given either on the command line or via the **-T** option. The default *N* is **1**.

--restrict

Disable the use of some potentially harmful options.

--sparse-version=MAJOR[.MINOR]

Set version of the sparse format to use (implies **--sparse**). This option implies **--sparse**. Valid argument values are **0.0**, **0.1**, and **1.0**. For a detailed discussion of sparse formats, refer to the **GNU Tar Manual**, appendix **D**, "Sparse Formats". Using **inf o** reader, it can be accessed running the following command: **info tar 'Sparse Formats'**.

-S, --sparse

Handle sparse files efficiently. Some files in the file system may have segments which were actually never written (quite often these are database files created by such systems as **DBM**). When given this option, **tar** attempts to determine if the file is sparse prior to archiving it, and if so, to reduce the resulting archive size by not dumping empty parts of the file.

Overwrite control

These options control **tar** actions when extracting a file over an existing copy on disk.

-k, --keep-old-files

Don't replace existing files when extracting.

--keep-newer-files

Don't replace existing files that are newer than their archive copies.

--keep-directory-symlink

Don't replace existing symlinks to directories when extracting.

--no-overwrite-dir

Preserve metadata of existing directories.

--one-top-level[=DIR]

Extract all files into *DIR*, or, if used without argument, into a subdirectory named by the base name of the archive (minus standard compression suffixes recognizable by **--auto-compress**).

--overwrite

Overwrite existing files when extracting.

--overwrite-dir

Overwrite metadata of existing directories when extracting (default).

--recursive-unlink

Recursively remove all files in the directory prior to extracting it.

--remove-files

Remove files from disk after adding them to the archive.

--skip-old-files

Don't replace existing files when extracting, silently skip over them.

-U, --unlink-first

Remove each file prior to extracting over it.

-W, --verify

Verify the archive after writing it.

Output stream selection**--ignore-command-error**

Ignore subprocess exit codes.

--no-ignore-command-error

Treat non-zero exit codes of children as error (default).

-O, --to-stdout

Extract files to standard output.

--to-command=COMMAND

Pipe extracted files to *COMMAND*. The argument is the pathname of an external program, optionally with command line arguments. The program will be invoked and the contents of the file being extracted supplied to it on its standard input. Additional data will be supplied via the following environment variables:

TAR_FILETYPE

Type of the file. It is a single letter with the following meaning:

f	Regular file
d	Directory
l	Symbolic link
h	Hard link
b	Block device
c	Character device

Currently only regular files are supported.

TAR_MODE

File mode, an octal number.

TAR_FILENAME

The name of the file.

TAR_REALNAME

Name of the file as stored in the archive.

TAR_UNAME

Name of the file owner.

TAR_GNAME

Name of the file owner group.

TAR_ATIME

Time of last access. It is a decimal number, representing seconds since the Epoch. If the archive provides times with nanosecond precision, the nanoseconds are appended to the timestamp after a decimal point.

TAR_MTIME

Time of last modification.

TAR_CTIME

Time of last status change.

TAR_SIZE

Size of the file.

TAR_UID

UID of the file owner.

TAR_GID

GID of the file owner.

Additionally, the following variables contain information about **tar** operation mode and the archive being processed:

TAR_VERSION

GNU **tar** version number.

TAR_ARCHIVE

The name of the archive **tar** is processing.

TAR_BLOCKING_FACTOR

Current blocking factor, i.e. number of 512-byte blocks in a record.

TAR_VOLUME

Ordinal number of the volume **tar** is processing (set if reading a multi-volume archive).

TAR_FORMAT

Format of the archive being processed. One of: **gnu**, **oldgnu**, **posix**, **ustar**, **v7**.

TAR_SUBCOMMAND

A short option (with a leading dash) describing the operation **tar** is executing.

Handling of file attributes**--atime-preserve[=METHOD]**

Preserve access times on dumped files, either by restoring the times after reading (*METHOD=replace*, this is the default) or by not setting the times in the first place (*METHOD=system*)

--delay-directory-restore

Delay setting modification times and permissions of extracted directories until the end of extraction. Use this option when extracting from an archive which has unusual member ordering.

--group=NAME[:GID]

Force *NAME* as group for added files. If *GID* is not supplied, *NAME* can be either a user name or numeric GID. In this case the missing part (GID or name) will be inferred from the current host's group database.

When used with **--group-map=FILE**, affects only those files whose owner group is not listed in

FILE.

--group-map=FILE

Read group translation map from *FILE*. Empty lines are ignored. Comments are introduced with # sign and extend to the end of line. Each non-empty line in *FILE* defines translation for a single group. It must consist of two fields, delimited by any amount of whitespace:

OLDGRP NEWGRP[:NEWGID]

OLDGRP is either a valid group name or a GID prefixed with +. Unless *NEWGID* is supplied, *NEWGRP* must also be either a valid group name or a +*GID*. Otherwise, both *NEWGRP* and *NEWGID* need not be listed in the system group database.

As a result, each input file with owner group *OLDGRP* will be stored in archive with owner group *NEWGRP* and GID *NEWGID*.

--mode=CHANGES

Force symbolic mode *CHANGES* for added files.

--mtime=DATE-OR-FILE

Set mtime for added files. *DATE-OR-FILE* is either a date/time in almost arbitrary format, or the name of an existing file. In the latter case the mtime of that file will be used.

-m, --touch

Don't extract file modified time.

--no-delay-directory-restore

Cancel the effect of the prior **--delay-directory-restore** option.

--no-same-owner

Extract files as yourself (default for ordinary users).

--no-same-permissions

Apply the user's umask when extracting permissions from the archive (default for ordinary users).

--numeric-owner

Always use numbers for user/group names.

--owner=NAME[:UID]

Force *NAME* as owner for added files. If *UID* is not supplied, *NAME* can be either a user name or numeric UID. In this case the missing part (UID or name) will be inferred from the current host's user database.

When used with **--owner-map=FILE**, affects only those files whose owner is not listed in *FILE*.

--owner-map=FILE

Read owner translation map from *FILE*. Empty lines are ignored. Comments are introduced with # sign and extend to the end of line. Each non-empty line in *FILE* defines translation for a single UID. It must consist of two fields, delimited by any amount of whitespace:

OLDUSR NEWUSR[:NEWUID]

OLDUSR is either a valid user name or a UID prefixed with +. Unless *NEWUID* is supplied, *NEWUSR* must also be either a valid user name or a +*UID*. Otherwise, both *NEWUSR* and *NEWUID* need not be listed in the system user database.

As a result, each input file owned by *OLDUSR* will be stored in archive with owner name *NEWUSR* and UID *NEWUID*.

-p, --preserve-permissions, --same-permissions

extract information about file permissions (default for superuser)

--same-owner

Try extracting files with the same ownership as exists in the archive (default for superuser).

-s, --preserve-order, --same-order

Sort names to extract to match archive

--sort=ORDER

When creating an archive, sort directory entries according to *ORDER*, which is one of **none**, **name**, or **inode**.

The default is **--sort=none**, which stores archive members in the same order as returned by the operating system.

Using **--sort=name** ensures the member ordering in the created archive is uniform and reproducible.

Using **--sort=inode** reduces the number of disk seeks made when creating the archive and thus can considerably speed up archivation. This sorting order is supported only if the underlying system provides the necessary information.

Extended file attributes

--acls Enable POSIX ACLs support.

--no-acls

Disable POSIX ACLs support.

--selinux

Enable SELinux context support.

--no-selinux

Disable SELinux context support.

--xattrs

Enable extended attributes support.

--no-xattrs

Disable extended attributes support.

--xattrs-exclude=PATTERN

Specify the exclude pattern for xattr keys. *PATTERN* is a POSIX regular expression, e.g. **--xattrs-exclude='^user.'**, to exclude attributes from the user namespace.

--xattrs-include=PATTERN

Specify the include pattern for xattr keys. *PATTERN* is a POSIX regular expression.

Device selection and switching**-f, --file=ARCHIVE**

Use archive file or device *ARCHIVE*. If this option is not given, **tar** will first examine the environment variable ‘TAPE’. If it is set, its value will be used as the archive name. Otherwise, **tar** will assume the compiled-in default. The default value can be inspected either using the **--show-defaults** option, or at the end of the **tar --help** output.

An archive name that has a colon in it specifies a file or device on a remote machine. The part before the colon is taken as the machine name or IP address, and the part after it as the file or device pathname, e.g.:

```
--file=remotehost:/dev/sr0
```

An optional username can be prefixed to the hostname, placing a @ sign between them.

By default, the remote host is accessed via the **rsh(1)** command. Nowadays it is common to use **ssh(1)** instead. You can do so by giving the following command line option:

```
--rsh-command=/usr/bin/ssh
```

The remote machine should have the **rmt(8)** command installed. If its pathname does not match **tar**'s default, you can inform **tar** about the correct pathname using the **--rmt-command** option.

--force-local

Archive file is local even if it has a colon.

-F, --info-script=COMMAND, --new-volume-script=COMMAND

Run *COMMAND* at the end of each tape (implies **-M**). The command can include arguments. When started, it will inherit **tar**'s environment plus the following variables:

TAR_VERSION

GNU **tar** version number.

TAR_ARCHIVE

The name of the archive **tar** is processing.

TAR_BLOCKING_FACTOR

Current blocking factor, i.e. number of 512-byte blocks in a record.

TAR_VOLUME

Ordinal number of the volume **tar** is processing (set if reading a multi-volume archive).

TAR_FORMAT

Format of the archive being processed. One of: **gnu**, **oldgnu**, **posix**, **ustar**, **v7**.

TAR_SUBCOMMAND

A short option (with a leading dash) describing the operation **tar** is executing.

TAR_FD

File descriptor which can be used to communicate the new volume name to **tar**.

If the info script fails, **tar** exits; otherwise, it begins writing the next volume.

-L, --tape-length=N

Change tape after writing *N*×1024 bytes. If *N* is followed by a size suffix (see the subsection **Size suffixes** below), the suffix specifies the multiplicative factor to be used instead of 1024.

This option implies **-M**.

-M, --multi-volume

Create/list/extract multi-volume archive.

--rmt-command=COMMAND

Use *COMMAND* instead of **rmt** when accessing remote archives. See the description of the **-f** option, above.

--rsh-command=COMMAND

Use *COMMAND* instead of **rsh** when accessing remote archives. See the description of the **-f** option, above.

--volno-file=FILE

When this option is used in conjunction with **--multi-volume**, **tar** will keep track of which volume of a multi-volume archive it is working in *FILE*.

Device blocking

-b, --blocking-factor=BLOCKS

Set record size to *BLOCKS*×512 bytes.

-B, --read-full-records

When listing or extracting, accept incomplete input records after end-of-file marker.

-i, --ignore-zeros

Ignore zeroed blocks in archive. Normally two consecutive 512-blocks filled with zeroes mean EOF and tar stops reading after encountering them. This option instructs it to read further and is useful when reading archives created with the **-A** option.

--record-size=NUMBER

Set record size. *NUMBER* is the number of bytes per record. It must be multiple of **512**. It can be suffixed with a **size suffix**, e.g. **--record-size=10K**, for 10 Kilobytes. See the subsection **Size suffixes**, for a list of valid suffixes.

Archive format selection**-H, --format=FORMAT**

Create archive of the given format. Valid formats are:

gnu GNU tar 1.13.x format

oldgnu GNU format as per tar <= 1.12.

pax, posix

POSIX 1003.1-2001 (pax) format.

ustar POSIX 1003.1-1988 (ustar) format.

v7 Old V7 tar format.

--old-archive, --portability

Same as **--format=v7**.

--pax-option=keyword[[[:]=value][,[keyword[[[:]=value]]...]

Control pax keywords when creating **PAX** archives (**-H pax**). This option is equivalent to the **-o** option of the **pax(1)** utility.

--posix

Same as **--format=posix**.

-V, --label=TEXT

Create archive with volume name *TEXT*. If listing or extracting, use *TEXT* as a globbing pattern for volume name.

Compression options**-a, --auto-compress**

Use archive suffix to determine the compression program.

-I, --use-compress-program=COMMAND

Filter data through *COMMAND*. It must accept the **-d** option, for decompression. The argument can contain command line options.

-j, --bzip2

Filter the archive through **bzip2(1)**.

-J, --xz

Filter the archive through **xz(1)**.

--lzip Filter the archive through **lzip(1)**.**--lzma**

Filter the archive through **lzma(1)**.

--lzop Filter the archive through **lzop(1)**.**--no-auto-compress**

Do not use archive suffix to determine the compression program.

-z, --gzip, --gunzip, --ungzip
 Filter the archive through **gzip(1)**.

-Z, --compress, --uncompress
 Filter the archive through **compress(1)**.
--zstd Filter the archive through **zstd(1)**.

Local file selection

--add-file=FILE
 Add *FILE* to the archive (useful if its name starts with a dash).

--backup[=CONTROL]
 Backup before removal. The *CONTROL* argument, if supplied, controls the backup policy. Its valid values are:

none, off
 Never make backups.

t, numbered
 Make numbered backups.

nil, existing
 Make numbered backups if numbered backups exist, simple backups otherwise.

never, simple
 Always make simple backups

If *CONTROL* is not given, the value is taken from the **VERSION_CONTROL** environment variable. If it is not set, **existing** is assumed.

-C, --directory=DIR

Change to *DIR* before performing any operations. This option is order-sensitive, i.e. it affects all options that follow.

--exclude=PATTERN

Exclude files matching *PATTERN*, a **glob(3)**-style wildcard pattern.

--exclude-backups

Exclude backup and lock files.

--exclude-caches

Exclude contents of directories containing file **CACHEDIR.TAG**, except for the tag file itself.

--exclude-caches-all

Exclude directories containing file **CACHEDIR.TAG** and the file itself.

--exclude-caches-under

Exclude everything under directories containing **CACHEDIR.TAG**

--exclude-ignore=FILE

Before dumping a directory, see if it contains *FILE*. If so, read exclusion patterns from this file. The patterns affect only the directory itself.

--exclude-ignore-recursive=FILE

Same as **--exclude-ignore**, except that patterns from *FILE* affect both the directory and all its subdirectories.

--exclude-tag=FILE

Exclude contents of directories containing *FILE*, except for *FILE* itself.

--exclude-tag-all=FILE

Exclude directories containing *FILE*.

- exclude-tag-under=FILE**
Exclude everything under directories containing *FILE*.
- exclude-vcs**
Exclude version control system directories.
- exclude-vcs-ignores**
Exclude files that match patterns read from VCS-specific ignore files. Supported files are: **.cvsignore**, **.gitignore**, **.bzrignore**, and **.hgignore**.
- h, --dereference**
Follow symlinks; archive and dump the files they point to.
- hard-dereference**
Follow hard links; archive and dump the files they refer to.
- K, --starting-file=MEMBER**
Begin at the given member in the archive.
- newer-mtime=DATE**
Work on files whose data changed after the *DATE*. If *D ATE* starts with / or . it is taken to be a file name; the mtime of that file is used as the date.
- no-null**
Disable the effect of the previous **--null** option.
- no-recursion**
Avoid descending automatically in directories.
- no-unquote**
Do not unquote input file or member names.
- no-verbatim-files-from**
Treat each line read from a file list as if it were supplied in the command line. I.e., leading and trailing whitespace is removed and, if the resulting string begins with a dash, it is treated as **tar** command line option.

This is the default behavior. The **--no-verbatim-files-from** option is provided as a way to restore it after **--verbatim-files-from** option.

This option is positional: it affects all **--files-from** options that occur after it in, until **--verbatim-files-from** option or end of line, whichever occurs first.

It is implied by the **--no-null** option.
--null Instruct subsequent **-T** options to read null-terminated names verbatim (disables special handling of names that start with a dash).

See also **--verbatim-files-from**.
- N, --newer=DATE, --after-date=DATE**
Only store files newer than *DATE*. If *D ATE* starts with / or . it is taken to be a file name; the mtime of that file is used as the date.
- one-file-system**
Stay in local file system when creating archive.
- P, --absolute-names**
Don't strip leading slashes from file names when creating archives.
- recursion**
Recurse into directories (default).

--suffix=STRING

Backup before removal, override usual suffix. Default suffix is `~`, unless overridden by environment variable `SIMPLE_BACKUP_SUFFIX`.

-T, --files-from=FILE

Get names to extract or create from *FILE*.

Unless specified otherwise, the *FILE* must contain a list of names separated by ASCII **LF** (i.e. one name per line). The names read are handled the same way as command line arguments. They undergo quote removal and word splitting, and any string that starts with a `-` is handled as **tar** command line option.

If this behavior is undesirable, it can be turned off using the **--verbatim-files-from** option.

The **--null** option instructs **tar** that the names in *FILE* are separated by ASCII **NUL** character, instead of **LF**. It is useful if the list is generated by **find(1) -print0** predicate.

--unquote

Unquote file or member names (default).

--verbatim-files-from

Treat each line obtained from a file list as a file name, even if it starts with a dash. File lists are supplied with the **--files-from** (**-T**) option. The default behavior is to handle names supplied in file lists as if they were typed in the command line, i.e. any names starting with a dash are treated as **tar** options. The **--verbatim-files-from** option disables this behavior.

This option affects all **--files-from** options that occur after it in the command line. Its effect is reverted by the **--no-verbatim-files-from** option.

This option is implied by the --null option.

See also **--add-file**.

-X, --exclude-from=FILE

Exclude files matching patterns listed in *FILE*.

File name transformations**--strip-components=NUMBER**

Strip *NUMBER* leading components from file names on extraction.

--transform=EXPRESSION, --xform=EXPRESSION

Use sed replace *EXPRESSION* to transform file names.

File name matching options

These options affect both exclude and include patterns.

--anchored

Patterns match file name start.

--ignore-case

Ignore case.

--no-anchored

Patterns match after any `/` (default for exclusion).

--no-ignore-case

Case sensitive matching (default).

--no-wildcards

Verbatim string matching.

--no-wildcards-match-slash

Wildcards do not match /.

--wildcards

Use wildcards (default for exclusion).

--wildcards-match-slash

Wildcards match / (default for exclusion).

Informative output**--checkpoint[=N]**

Display progress messages every *N*th record (default 10).

--checkpoint-action=ACTION

Run *ACTION* on each checkpoint.

--clamp-mtime

Only set time when the file is more recent than what was given with --mtime.

--full-time

Print file time to its full resolution.

--index-file=FILE

Send verbose output to *FILE*.

-l, --check-links

Print a message if not all links are dumped.

--no-quote-chars=STRING

Disable quoting for characters from *STRING*.

--quote-chars=STRING

Additionally quote characters from *STRING*.

--quoting-style=STYLE

Set quoting style for file and member names. Valid values for *STYLE* are **literal**, **shell**, **shell-always**, **c**, **c-maybe**, **escape**, **locale**, **clocale**.

-R, --block-number

Show block number within archive with each message.

--show-omitted-dirs

When listing or extracting, list each directory that does not match search criteria.

--show-transformed-names, --show-stored-names

Show file or archive names after transformation by --strip and --transform options.

--totals[=SIGNAL]

Print total bytes after processing the archive. If *SIGNAL* is given, print total bytes when this signal is delivered. Allowed signals are: **SIGHUP**, **SIGQUIT**, **SIGINT**, **SIGUSR1**, and **SIGUSR2**. The **SIG** prefix can be omitted.

--utc

Print file modification times in UTC.

-v, --verbose

Verbosely list files processed. Each instance of this option on the command line increases the verbosity level by one. The maximum verbosity level is 3. For a detailed discussion of how various verbosity levels affect tar's output, please refer to **GNU Tar Manual**, subsection 2.5.1 "The --verbose Option".

--warning=KEYWORD

Enable or disable warning messages identified by *KEYWORD*. The messages are suppressed if *KEYWORD* is prefixed with **no-** and enabled otherwise.

Multiple --warning messages accumulate.

Keywords controlling general **tar** operation:

all Enable all warning messages. This is the default.

none Disable all warning messages.

filename-with-nuls

"%s: file name read contains nul character"

alone-zero-block

"A lone zero block at %s"

Keywords applicable for **tar --create**:

cachedir

"%s: contains a cache directory tag %s; %s"

file-shrank

"%s: File shrank by %s bytes; padding with zeros"

xdev "%s: file is on a different filesystem; not dumped"

file-ignored

"%s: Unknown file type; file ignored"

"%s: socket ignored"

"%s: door ignored"

file-unchanged

"%s: file is unchanged; not dumped"

ignore-archive

"%s: file is the archive; not dumped"

file-removed

"%s: File removed before we read it"

file-changed

"%s: file changed as we read it"

failed-read

Suppresses warnings about unreadable files or directories. This keyword applies only if used together with the **--ignore-failed-read** option.

Keywords applicable for **tar --extract**:

existing-file

"%s: skipping existing file"

timestamp

"%s: implausibly old time stamp %s"

"%s: time stamp %s is %s s in the future"

contiguous-cast

"Extracting contiguous files as regular files"

symlink-cast

"Attempting extraction of symbolic links as hard links"

unknown-cast

"%s: Unknown file type '%c', extracted as normal file"

ignore-newer

"Current %s is newer or same age"

unknown-keyword

"Ignoring unknown extended header keyword '%s'"

decompress-program

Controls verbose description of failures occurring when trying to run alternative decompressor programs. This warning is disabled by default (unless **--verbose** is used). A common example of what you can get when using this warning is:

```
$ tar --warning=decompress-program -x -f archive.Z
tar (child): cannot run compress: No such file or directory
tar (child): trying gzip
```

This means that **tar** first tried to decompress **archive.Z** using **compress**, and, when that failed, switched to **gzip**.

record-size

"Record size = %lu blocks"

Keywords controlling incremental extraction:

rename-directory

"%s: Directory has been renamed from %s"
"%s: Directory has been renamed"

new-directory

"%s: Directory is new"

xdev

"%s: directory is on a different device: not purging"

bad-dumpdir

"Malformed dumpdir: 'X' never used"

-w, --interactive, --confirmation

Ask for confirmation for every action.

Compatibility options

-o When creating, same as **--old-archive**. When extracting, same as **--no-same-owner**.

Size suffixes

<i>Suffix</i>	<i>Units</i>	<i>Byte Equivalent</i>
b	Blocks	<i>SIZE</i> x 512
B	Kilobytes	<i>SIZE</i> x 1024
c	Bytes	<i>SIZE</i>
G	Gigabytes	<i>SIZE</i> x 1024^3
K	Kilobytes	<i>SIZE</i> x 1024
k	Kilobytes	<i>SIZE</i> x 1024
M	Megabytes	<i>SIZE</i> x 1024^2
P	Petabytes	<i>SIZE</i> x 1024^5
T	Terabytes	<i>SIZE</i> x 1024^4
w	Words	<i>SIZE</i> x 2

RETURN VALUE

Tar exit code indicates whether it was able to successfully perform the requested operation, and if not, what kind of error occurred.

- 0** Successful termination.
- 1** *Some files differ*: If tar was invoked with the **--compare** (**--diff**, **-d**) command line option, this means that some files in the archive differ from their disk counterparts. If tar was given one of the **--create**, **--append** or **--update** options, this exit code means that some files were changed while being archived and so the resulting archive does not contain the exact copy of the file set.
- 2** *Fatal error*: This means that some fatal, unrecoverable error occurred.

If a subprocess that had been invoked by **tar** exited with a nonzero exit code, **tar** itself exits with that code as well. This can happen, for example, if a compression option (e.g. **-z**) was used and the external

compressor program failed. Another example is **rmt** failure during backup to a remote device.

SEE ALSO

bzip2(1), **compress(1)**, **gzip(1)**, **lzma(1)**, **lzop(1)**, **rmt(8)**, **symlink(7)**, **xz(1)**, **zstd(1)**.

Complete **tar** manual: run **info tar** or use **emacs(1)** info mode to read it.

Online copies of **GNU tar** documentation in various formats can be found at:

<http://www.gnu.org/software/tar/manual>

BUG REPORTS

Report bugs to <bug-tar@gnu.org>.

COPYRIGHT

Copyright © 2013-2019 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

NAME

tee – read from standard input and write to standard output and files

SYNOPSIS

tee [*OPTION*]... [*FILE*]...

DESCRIPTION

Copy standard input to each FILE, and also to standard output.

-a, --append

append to the given FILES, do not overwrite

-i, --ignore-interrupts

ignore interrupt signals

-p diagnose errors writing to non pipes

--output-error[=*MODE*]

set behavior on write error. See MODE below

--help display this help and exit

--version

output version information and exit

MODE determines behavior with write errors on the outputs:

'warn' diagnose errors writing to any output

'warn-nopipe'

diagnose errors writing to any output not a pipe

'exit' exit on error writing to any output

'exit-nopipe'

exit on error writing to any output not a pipe

The default MODE for the **-p** option is 'warn-nopipe'. The default operation when **--output-error** is not specified, is to exit immediately on error writing to a pipe, and diagnose errors writing to non pipe outputs.

AUTHOR

Written by Mike Parker, Richard M. Stallman, and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/tee>>

or available locally via: info '(coreutils) tee invocation'

NAME

tee – duplicating pipe content

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#define _GNU_SOURCE      /* See feature_test_macros(7) */
#include <fcntl.h>
ssize_t tee(int fd_in, int fd_out, size_t len, unsigned int flags);
```

DESCRIPTION

tee() duplicates up to *len* bytes of data from the pipe referred to by the file descriptor *fd_in* to the pipe referred to by the file descriptor *fd_out*. It does not consume the data that is duplicated from *fd_in*; therefore, that data can be copied by a subsequent **splice(2)**.

flags is a bit mask that is composed by ORing together zero or more of the following values:

SPLICE_F_MOVE	Currently has no effect for tee() ; see splice(2) .
SPLICE_F_NONBLOCK	Do not block on I/O; see splice(2) for further details.
SPLICE_F_MORE	Currently has no effect for tee() , but may be implemented in the future; see splice(2) .
SPLICE_F_GIFT	Unused for tee() ; see vmsplice(2) .

RETURN VALUE

Upon successful completion, **tee()** returns the number of bytes that were duplicated between the input and output. A return value of 0 means that there was no data to transfer, and it would not make sense to block, because there are no writers connected to the write end of the pipe referred to by *fd_in*.

On error, **tee()** returns -1 and *errno* is set to indicate the error.

ERRORS

EAGAIN

SPLICE_F_NONBLOCK was specified in *flags* or one of the file descriptors had been marked as nonblocking (**O_NONBLOCK**), and the operation would block.

EINVAL

fd_in or *fd_out* does not refer to a pipe; or *fd_in* and *fd_out* refer to the same pipe.

ENOMEM

Out of memory.

VERSIONS

The **tee()** system call first appeared in Linux 2.6.17; library support was added in glibc 2.5.

STANDARDS

This system call is Linux-specific.

NOTES

Conceptually, **tee()** copies the data between the two pipes. In reality no real data copying takes place though: under the covers, **tee()** assigns data to the output by merely grabbing a reference to the input.

EXAMPLES

The example below implements a basic **tee(1)** program using the **tee()** system call. Here is an example of its use:

```
$ date | ./a.out out.log | cat
Tue Oct 28 10:06:00 CET 2014
$ cat out.log
Tue Oct 28 10:06:00 CET 2014
```

Program source

```

#define _GNU_SOURCE
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
    int      fd;
    ssize_t  len, slen;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    for (;;) {
        /*
         * tee stdin to stdout.
         */
        len = tee(STDIN_FILENO, STDOUT_FILENO,
                  INT_MAX, SPLICE_F_NONBLOCK);
        if (len < 0) {
            if (errno == EAGAIN)
                continue;
            perror("tee");
            exit(EXIT_FAILURE);
        }
        if (len == 0)
            break;

        /*
         * Consume stdin by splicing it to a file.
         */
        while (len > 0) {
            slen = splice(STDIN_FILENO, NULL, fd, NULL,
                          len, SPLICE_F_MOVE);
            if (slen < 0) {
                perror("splice");
                exit(EXIT_FAILURE);
            }
            len -= slen;
        }
    }
}

```

```
    }  
  
    close(fd);  
    exit(EXIT_SUCCESS);  
}
```

SEE ALSO

[splice\(2\)](#), [vmsplice\(2\)](#), [pipe\(7\)](#)

NAME

tkill, **tkill** – send a signal to a thread

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <signal.h>      /* Definition of SIG* constants */
#include <sys/syscall.h>   /* Definition of SYS_* constants */
#include <unistd.h>

[[deprecated]] int syscall(SYS_tkill, pid_t tid, int sig);

#include <signal.h>

int tkill(pid_t tgid, pid_t tid, int sig);
```

Note: glibc provides no wrapper for **tkill()**, necessitating the use of **syscall(2)**.

DESCRIPTION

tgkill() sends the signal *sig* to the thread with the thread ID *tid* in the thread group *tgid*. (By contrast, **kill(2)** can be used to send a signal only to a process (i.e., thread group) as a whole, and the signal will be delivered to an arbitrary thread within that process.)

tkill() is an obsolete predecessor to **tgkill()**. It allows only the target thread ID to be specified, which may result in the wrong thread being signaled if a thread terminates and its thread ID is recycled. Avoid using this system call.

These are the raw system call interfaces, meant for internal thread library use.

RETURN VALUE

On success, zero is returned. On error, **-1** is returned, and *errno* is set to indicate the error.

ERRORS

EAGAIN

The **RLIMIT_SIGPENDING** resource limit was reached and *sig* is a real-time signal.

EAGAIN

Insufficient kernel memory was available and *sig* is a real-time signal.

EINVAL

An invalid thread ID, thread group ID, or signal was specified.

EPERM

Permission denied. For the required permissions, see **kill(2)**.

ESRCH

No process with the specified thread ID (and thread group ID) exists.

VERSIONS

tkill() is supported since Linux 2.4.19 / 2.5.4. **tgkill()** was added in Linux 2.5.75.

Library support for **tgkill()** was added in glibc 2.30.

STANDARDS

tkill() and **tgkill()** are Linux-specific and should not be used in programs that are intended to be portable.

NOTES

See the description of **CLONE_THREAD** in **clone(2)** for an explanation of thread groups.

Before glibc 2.30, there was also no wrapper function for **tgkill()**.

SEE ALSO

clone(2), **gettid(2)**, **kill(2)**, **rt_sigqueueinfo(2)**

NAME

time – time a simple command or give resource usage

SYNOPSIS

time [*options*] *command* [*arguments...*]

DESCRIPTION

The **time** command runs the specified program *command* with the given arguments. When *command* finishes, **time** writes a message to standard error giving timing statistics about this program run. These statistics consist of (i) the elapsed real time between invocation and termination, (ii) the user CPU time (the sum of the *tms_utime* and *tms_cutime* values in a *struct tms* as returned by **times(2)**), and (iii) the system CPU time (the sum of the *tms_stime* and *tms_cstime* values in a *struct tms* as returned by **times(2)**).

Note: some shells (e.g., **bash(1)**) have a built-in **time** command that provides similar information on the usage of time and possibly other resources. To access the real command, you may need to specify its pathname (something like */usr/bin/time*).

OPTIONS

-p When in the POSIX locale, use the precise traditional format

```
"real %f\nuser %f\nsys %f\n"
```

(with numbers in seconds) where the number of decimals in the output for %f is unspecified but is sufficient to express the clock tick accuracy, and at least one.

EXIT STATUS

If *command* was invoked, the exit status is that of *command*. Otherwise, it is 127 if *command* could not be found, 126 if it could be found but could not be invoked, and some other nonzero value (1–125) if something else went wrong.

ENVIRONMENT

The variables **LANG**, **LC_ALL**, **LC_CTYPE**, **LC_MESSAGES**, **LC_NUMERIC**, and **NLSPATH** are used for the text and formatting of the output. **PATH** is used to search for *command*.

GNU VERSION

Below a description of the GNU 1.7 version of **time**. Disregarding the name of the utility, GNU makes it output lots of useful information, not only about time used, but also on other resources like memory, I/O and IPC calls (where available). The output is formatted using a format string that can be specified using the **-f** option or the **TIME** environment variable.

The default format string is:

```
%User %Ssystem %Eelapsed %PCPU (%Xtext+%Ddata %Mmax)k
%Iinputs+%Ooutputs (%Fmajor+%Rminor)pagefaults %Wswaps
```

When the **-p** option is given, the (portable) output format is used:

```
real %e
user %U
sys %S
```

The format string

The format is interpreted in the usual printf-like way. Ordinary characters are directly copied, tab, newline, and backslash are escaped using \t, \n, and \\, a percent sign is represented by %% , and otherwise % indicates a conversion. The program **time** will always add a trailing newline itself. The conversions follow. All of those used by **tcsh(1)** are supported.

Time

%E Elapsed real time (in [hours:]minutes:seconds).

%e (Not in **tcsh(1)**.) Elapsed real time (in seconds).

%S Total number of CPU-seconds that the process spent in kernel mode.

- %U** Total number of CPU-seconds that the process spent in user mode.
%P Percentage of the CPU that this job got, computed as $(\%U + \%S) / \%E$.

Memory

- %M** Maximum resident set size of the process during its lifetime, in Kbytes.
%t (Not in **tcsesh(1)**.) Average resident set size of the process, in Kbytes.
%K Average total (data+stack+text) memory use of the process, in Kbytes.
%D Average size of the process's unshared data area, in Kbytes.
%p (Not in **tcsesh(1)**.) Average size of the process's unshared stack space, in Kbytes.
%X Average size of the process's shared text space, in Kbytes.
%Z (Not in **tcsesh(1)**.) System's page size, in bytes. This is a per-system constant, but varies between systems.
%F Number of major page faults that occurred while the process was running. These are faults where the page has to be read in from disk.
%R Number of minor, or recoverable, page faults. These are faults for pages that are not valid but which have not yet been claimed by other virtual pages. Thus the data in the page is still valid but the system tables must be updated.
%W Number of times the process was swapped out of main memory.
%c Number of times the process was context-switched involuntarily (because the time slice expired).
%w Number of waits: times that the program was context-switched voluntarily, for instance while waiting for an I/O operation to complete.

I/O

- %I** Number of filesystem inputs by the process.
%O Number of filesystem outputs by the process.
%r Number of socket messages received by the process.
%s Number of socket messages sent by the process.
%k Number of signals delivered to the process.
%C (Not in **tcsesh(1)**.) Name and command-line arguments of the command being timed.
%x (Not in **tcsesh(1)**.) Exit status of the command.

GNU options

- f** *format*, **--format**=*format*
 Specify output format, possibly overriding the format specified in the environment variable TIME.
- p**, **--portability**
 Use the portable output format.
- o** *file*, **--output**=*file*
 Do not send the results to *stderr*, but overwrite the specified file.
- a**, **--append**
 (Used together with -o.) Do not overwrite but append.
- v**, **--verbose**
 Give very verbose output about all the program knows about.
- q**, **--quiet**
 Don't report abnormal program termination (where *command* is terminated by a signal) or non-zero exit status.

GNU standard options

--help Print a usage message on standard output and exit successfully.

-V, --version

Print version information on standard output, then exit successfully.

-- Terminate option list.

BUGS

Not all resources are measured by all versions of UNIX, so some of the values might be reported as zero. The present selection was mostly inspired by the data provided by 4.2 or 4.3BSD.

GNU time version 1.7 is not yet localized. Thus, it does not implement the POSIX requirements.

The environment variable **TIME** was badly chosen. It is not unusual for systems like **autoconf(1)** or **make(1)** to use environment variables with the name of a utility to override the utility to be used. Uses like MORE or TIME for options to programs (instead of program pathnames) tend to lead to difficulties.

It seems unfortunate that **-o** overwrites instead of appends. (That is, the **-a** option should be the default.)

Mail suggestions and bug reports for GNU **time** to bug-time@gnu.org. Please include the version of **time**, which you can get by running

```
time --version
```

and the operating system and C compiler you used.

SEE ALSO

bash(1), tcsh(1), times(2), wait3(2)

NAME

`time` – get time in seconds

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <time.h>
time_t time(time_t *_Nullable tloc);
```

DESCRIPTION

`time()` returns the time as the number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).

If *tloc* is non-NULL, the return value is also stored in the memory pointed to by *tloc*.

RETURN VALUE

On success, the value of time in seconds since the Epoch is returned. On error, $((\text{time_t}) - 1)$ is returned, and *errno* is set to indicate the error.

ERRORS

EFAULT

tloc points outside your accessible address space (but see BUGS).

On systems where the C library `time()` wrapper function invokes an implementation provided by the `vdso(7)` (so that there is no trap into the kernel), an invalid address may instead trigger a `SIGSEGV` signal.

STANDARDS

SVr4, 4.3BSD, C99, POSIX.1-2001. POSIX does not specify any error conditions.

NOTES

POSIX.1 defines *seconds since the Epoch* using a formula that approximates the number of seconds between a specified time and the Epoch. This formula takes account of the facts that all years that are evenly divisible by 4 are leap years, but years that are evenly divisible by 100 are not leap years unless they are also evenly divisible by 400, in which case they are leap years. This value is not the same as the actual number of seconds between the time and the Epoch, because of leap seconds and because system clocks are not required to be synchronized to a standard reference. The intention is that the interpretation of seconds since the Epoch values be consistent; see POSIX.1-2008 Rationale A.4.15 for further rationale.

On Linux, a call to `time()` with *tloc* specified as NULL cannot fail with the error `EOVERFLOW`, even on ABIs where *time_t* is a signed 32-bit integer and the clock reaches or exceeds 2**31 seconds (2038-01-19 03:14:08 UTC, ignoring leap seconds). (POSIX.1 permits, but does not require, the `EOVERFLOW` error in the case where the seconds since the Epoch will not fit in *time_t*.) Instead, the behavior on Linux is undefined when the system time is out of the *time_t* range. Applications intended to run after 2038 should use ABIs with *time_t* wider than 32 bits.

BUGS

Error returns from this system call are indistinguishable from successful reports that the time is a few seconds *before* the Epoch, so the C library wrapper function never sets *errno* as a result of this call.

The *tloc* argument is obsolescent and should always be NULL in new code. When *tloc* is NULL, the call cannot fail.

C library/kernel differences

On some architectures, an implementation of `time()` is provided in the `vdso(7)`.

SEE ALSO

`date(1)`, `gettimeofday(2)`, `ctime(3)`, `ftime(3)`, `time(7)`, `vdso(7)`

NAME

`time` – time functions for gawk

SYNOPSIS

```
@load "time"

time = gettimeofday()
ret = sleep(amount)
```

CAUTION

This extension is deprecated in favor of the **timex** extension in the *gawkextlib* project. In the next major release of *gawk*, loading it will issue a warning. It will be removed from the *gawk* distribution in the major release after the next one.

DESCRIPTION

The *time* extension adds two functions named **gettimeofday()** and **sleep()**, as follows.

gettimeofday()

This function returns the number of seconds since the Epoch as a floating-point value. It should have subsecond precision. It returns `-1` upon error and sets **ERRNO** to indicate the problem.

sleep(seconds)

This function attempts to sleep for the given amount of seconds, which may include a fractional portion. If *seconds* is negative, or the attempt to sleep fails, then it returns `-1` and sets **ERRNO**. Otherwise, the function should return `0` after sleeping for the indicated amount of time.

EXAMPLE

```
@load "time"
...
printf "It is now %g seconds since the Epoch\n", gettimeofday()
printf "Pausing for a while... " ; sleep(2.5) ; print "done"
```

SEE ALSO

GAWK: Effective AWK Programming, *filefuncs(3am)*, *fnmatch(3am)*, *fork(3am)*, *inplace(3am)*, *ordchr(3am)*, *readdir(3am)*, *readfile(3am)*, *revoutput(3am)*, *rwarray(3am)*.

gettimeofday(2), *nanosleep(2)*, *select(2)*.

AUTHOR

Arnold Robbins, arnold@skeeve.com.

COPYING PERMISSIONS

Copyright © 2012, 2013, 2018, Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this manual page provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual page under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual page into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Foundation.

NAME

time – overview of time and timers

DESCRIPTION

Real time and process time

Real time is defined as time measured from some fixed point, either from a standard point in the past (see the description of the Epoch and calendar time below), or from some point (e.g., the start) in the life of a process (*elapsed time*).

Process time is defined as the amount of CPU time used by a process. This is sometimes divided into *user* and *system* components. User CPU time is the time spent executing code in user mode. System CPU time is the time spent by the kernel executing in system mode on behalf of the process (e.g., executing system calls). The **time**(1) command can be used to determine the amount of CPU time consumed during the execution of a program. A program can determine the amount of CPU time it has consumed using **times**(2), **getrusage**(2), or **clock**(3).

The hardware clock

Most computers have a (battery-powered) hardware clock which the kernel reads at boot time in order to initialize the software clock. For further details, see **rtc**(4) and **hwclock**(8).

The software clock, HZ, and jiffies

The accuracy of various system calls that set timeouts, (e.g., **select**(2), **sigtimedwait**(2)) and measure CPU time (e.g., **getrusage**(2)) is limited by the resolution of the *software clock*, a clock maintained by the kernel which measures time in *jiffies*. The size of a jiffy is determined by the value of the kernel constant **HZ**.

The value of **HZ** varies across kernel versions and hardware platforms. On i386 the situation is as follows: on kernels up to and including Linux 2.4.x, **HZ** was 100, giving a jiffy value of 0.01 seconds; starting with Linux 2.6.0, **HZ** was raised to 1000, giving a jiffy of 0.001 seconds. Since Linux 2.6.13, the **HZ** value is a kernel configuration parameter and can be 100, 250 (the default) or 1000, yielding a jiffies value of, respectively, 0.01, 0.004, or 0.001 seconds. Since Linux 2.6.20, a further frequency is available: 300, a number that divides evenly for the common video frame rates (PAL, 25 Hz; NTSC, 30 Hz).

The **times**(2) system call is a special case. It reports times with a granularity defined by the kernel constant **USER_HZ**. User-space applications can determine the value of this constant using **sysconf(_SC_CLK_TCK)**.

System and process clocks; time namespaces

The kernel supports a range of clocks that measure various kinds of elapsed and virtual (i.e., consumed CPU) time. These clocks are described in **clock_gettime**(2). A few of the clocks are settable using **clock_settime**(2). The values of certain clocks are virtualized by time namespaces; see **time_namespaces**(7).

High-resolution timers

Before Linux 2.6.21, the accuracy of timer and sleep system calls (see below) was also limited by the size of the jiffy.

Since Linux 2.6.21, Linux supports high-resolution timers (HRTs), optionally configurable via **CONFIG_HIGH_RES_TIMERS**. On a system that supports HRTs, the accuracy of sleep and timer system calls is no longer constrained by the jiffy, but instead can be as accurate as the hardware allows (microsecond accuracy is typical of modern hardware). You can determine whether high-resolution timers are supported by checking the resolution returned by a call to **clock_getres**(2) or looking at the "resolution" entries in **/proc/timer_list**.

HRTs are not supported on all hardware architectures. (Support is provided on x86, ARM, and PowerPC, among others.)

The Epoch

UNIX systems represent time in seconds since the *Epoch*, 1970-01-01 00:00:00 +0000 (UTC).

A program can determine the *calendar time* via the **clock_gettime**(2) **CLOCK_REALTIME** clock, which returns time (in seconds and nanoseconds) that have elapsed since the Epoch; **time**(2) provides similar information, but only with accuracy to the nearest second. The system time can be changed using

clock_settime(2).

Broken-down time

Certain library functions use a structure of type *tm* to represent *broken-down time*, which stores time value separated out into distinct components (year, month, day, hour, minute, second, etc.). This structure is described in **tm(3type)**, which also describes functions that convert between calendar time and broken-down time. Functions for converting between broken-down time and printable string representations of the time are described in **ctime(3)**, **strftime(3)**, and **strptime(3)**.

Sleeping and setting timers

Various system calls and functions allow a program to sleep (suspend execution) for a specified period of time; see **nanosleep(2)**, **clock_nanosleep(2)**, and **sleep(3)**.

Various system calls allow a process to set a timer that expires at some point in the future, and optionally at repeated intervals; see **alarm(2)**, **getitimer(2)**, **timerfd_create(2)**, and **timer_create(2)**.

Timer slack

Since Linux 2.6.28, it is possible to control the "timer slack" value for a thread. The timer slack is the length of time by which the kernel may delay the wake-up of certain system calls that block with a timeout. Permitting this delay allows the kernel to coalesce wake-up events, thus possibly reducing the number of system wake-ups and saving power. For more details, see the description of **PR_SET_TIMERSLACK** in **prctl(2)**.

SEE ALSO

date(1), **time(1)**, **timeout(1)**, **adjtimex(2)**, **alarm(2)**, **clock_gettime(2)**, **clock_nanosleep(2)**, **getitimer(2)**, **getrlimit(2)**, **getrusage(2)**, **gettimeofday(2)**, **nanosleep(2)**, **stat(2)**, **time(2)**, **timer_create(2)**, **timerfd_create(2)**, **times(2)**, **utime(2)**, **adjtime(3)**, **clock(3)**, **clock_getcpuclockid(3)**, **ctime(3)**, **ntp_adjtime(3)**, **ntp_gettime(3)**, **pthread_getcpuclockid(3)**, **sleep(3)**, **strftime(3)**, **strptime(3)**, **timeradd(3)**, **usleep(3)**, **rtc(4)**, **time_namespaces(7)**, **hwclock(8)**

NAME

`time.conf` – configuration file for the pam_time module

DESCRIPTION

The pam_time PAM module does not authenticate the user, but instead it restricts access to a system and/or specific applications at various times of the day and on specific days or over various terminal lines. This module can be configured to deny access to (individual) users based on their name, the time of day, the day of week, the service they are applying for and their terminal from which they are making their request.

For this module to function correctly there must be a correctly formatted /etc/security/time.conf file present. White spaces are ignored and lines maybe extended with '\' (escaped newlines). Text following a '#' is ignored to the end of the line.

The syntax of the lines is as follows:

services;ttys;users;times

In words, each rule occupies a line, terminated with a newline or the beginning of a comment; a '#'. It contains four fields separated with semicolons, ';':

The first field, the *services* field, is a logic list of PAM service names that the rule applies to.

The second field, the *tty* field, is a logic list of terminal names that this rule applies to.

The third field, the *users* field, is a logic list of users or a netgroup of users to whom this rule applies.

A logic list namely means individual tokens that are optionally prefixed with '!' (logical not) and separated with '&' (logical and) and '|' (logical or).

For these items the simple wildcard '*' may be used only once. With netgroups no wildcards or logic operators are allowed.

The *times* field is used to indicate the times at which this rule applies. The format here is a logic list of day/time-range entries. The days are specified by a sequence of two character entries, MoTuSa for example is Monday Tuesday and Saturday. Note that repeated days are unset MoMo = no day, and MoWk = all weekdays bar Monday. The two character combinations accepted are Mo Tu We Th Fr Sa Su Wk Wd Al, the last two being week-end days and all 7 days of the week respectively. As a final example, AlFr means all days except Friday.

Each day/time-range can be prefixed with a '!' to indicate "anything but". The time-range part is two 24-hour times HHMM, separated by a hyphen, indicating the start and finish time (if the finish time is smaller than the start time it is deemed to apply on the following day).

For a rule to be active, ALL of service+ttys+users must be satisfied by the applying process.

Note, currently there is no daemon enforcing the end of a session. This needs to be remedied.

Poorly formatted rules are logged as errors using `syslog(3)`.

EXAMPLES

These are some example lines which might be specified in /etc/security/time.conf.

All users except for *root* are denied access to console-login at all times:

```
login ; tty* & !typ* ; !root ; !Al0000-2400
```

Games (configured to use PAM) are only to be accessed out of working hours. This rule does not apply to the user *waster*:

```
games ; * ; !waster ; Wd0000-2400 | Wk1800-0800
```

SEE ALSO

pam_time(8), pam.d(5), pam(7)

AUTHOR

pam_time was written by Andrew G. Morgan <morgan@kernel.org>.

NAME

`system_data_types` – overview of system data types

DESCRIPTION

aiocb

Include: `<aio.h>`.

```
struct aiocb {
    int             aio_fildes;      /* File descriptor */
    off_t           aio_offset;     /* File offset */
    volatile void * aio_buf;        /* Location of buffer */
    size_t          aio_nbytes;     /* Length of transfer */
    int             aio_reqprio;    /* Request priority offset */
    struct sigevent aio_sigevent;  /* Signal number and value */
    int             aio_lio_opcode; /* Operation to be performed */
};
```

For further information about this structure, see [aio\(7\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [aio_cancel\(3\)](#), [aio_error\(3\)](#), [aio_fsync\(3\)](#), [aio_read\(3\)](#), [aio_return\(3\)](#), [aio_suspend\(3\)](#), [aio_write\(3\)](#), [lio_listio\(3\)](#)

clock_t

Include: `<time.h>` or `<sys/types.h>`. Alternatively, `<sys/time.h>`.

Used for system time in clock ticks or **CLOCKS_PER_SEC** (defined in `<time.h>`). According to POSIX, it shall be an integer type or a real-floating type.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [times\(2\)](#), [clock\(3\)](#)

clockid_t

Include: `<sys/types.h>`. Alternatively, `<time.h>`.

Used for clock ID type in the clock and timer functions. According to POSIX, it shall be defined as an arithmetic type.

Conforming to: POSIX.1-2001 and later.

See also: [clock_adjtime\(2\)](#), [clock_getres\(2\)](#), [clock_nanosleep\(2\)](#), [timer_create\(2\)](#), [clock_getcpuclockid\(3\)](#)

dev_t

Include: `<sys/types.h>`. Alternatively, `<sys/stat.h>`.

Used for device IDs. According to POSIX, it shall be an integer type. For further details of this type, see [makedev\(3\)](#).

Conforming to: POSIX.1-2001 and later.

See also: [mknod\(2\)](#), [stat\(2\)](#)

div_t

Include: `<stdlib.h>`.

```
typedef struct {
    int quot; /* Quotient */
    int rem; /* Remainder */
} div_t;
```

It is the type of the value returned by the [div\(3\)](#) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: [div\(3\)](#)

double_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *double*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *double_t* is *double*.
- 1 *double_t* is *double*.
- 2 *double_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *double_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *float_t* type in this page.

fd_set

Include: <sys/select.h>. Alternatively, <sys/time.h>.

A structure type that can represent a set of file descriptors. According to POSIX, the maximum number of file descriptors in an *fd_set* structure is the value of the macro **FD_SETSIZE**.

Conforming to: POSIX.1-2001 and later.

See also: **select**(2)

fenv_t

Include: <fenv.h>.

This type represents the entire floating-point environment, including control modes and status flags; for further details, see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

fexcept_t

Include: <fenv.h>.

This type represents the floating-point status flags collectively; for further details see **fenv**(3).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fenv**(3)

FILE

Include: <stdio.h>. Alternatively, <wchar.h>.

An object type used for streams.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **fclose**(3), **flockfile**(3), **fopen**(3), **fprintf**(3), **fread**(3), **fscanf**(3), **stdin**(3), **stdio**(3)

float_t

Include: <math.h>.

The implementation's most efficient floating type at least as wide as *float*. Its type depends on the value of the macro **FLT_EVAL_METHOD** (defined in <float.h>):

- 0 *float_t* is *float*.
- 1 *float_t* is *double*.
- 2 *float_t* is *long double*.

For other values of **FLT_EVAL_METHOD**, the type of *float_t* is implementation-defined.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *double_t* type in this page.

gid_t

Include: <sys/types.h>. Alternatively, <grp.h>, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold group IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: chown(2), getgid(2), getegid(2), getgroups(2), getresgid(2), getgrnam(2), credentials(7)

id_t

Include: <sys/types.h>. Alternatively, <sys/resource.h>.

A type used to hold a general identifier. According to POSIX, this shall be an integer type that can be used to contain a *pid_t*, *uid_t*, or *gid_t*.

Conforming to: POSIX.1-2001 and later.

See also: getpriority(2), waitid(2)

imaxdiv_t

Include: <inttypes.h>.

```
typedef struct {
    intmax_t      quot; /* Quotient */
    intmax_t      rem;   /* Remainder */
} imaxdiv_t;
```

It is the type of the value returned by the imaxdiv(3) function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: imaxdiv(3)

intmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

A signed integer type capable of representing any value of any signed integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [INTMAX_MIN, INTMAX_MAX].

The macro INTMAX_C() expands its argument to an integer constant of type *intmax_t*.

The length modifier for *intmax_t* for the printf(3) and the scanf(3) families of functions is **j**; resulting commonly in %**jd** or %**ji** for printing *intmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *intmax_t* is not large enough to represent values of type __int128 in implementations where __int128 is defined and long long is less than 128 bits wide.

See also: the *uintmax_t* type in this page.

intN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

int8_t, *int16_t*, *int32_t*, *int64_t*

A signed integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [INTN_MIN, INTN_MAX], substituting N by the appropriate number.

According to POSIX, *int8_t*, *int16_t*, and *int32_t* are required; *int64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *intN_t* types for the printf(3) family of functions are expanded by macros of the forms **PRIIdN** and **PRIiN** (defined in <inttypes.h>); resulting for example in %"**PRI64**" or %"**PRIi64**" for printing *int64_t* values. The length modifiers for the *intN_t*

types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNdN** and **SCNiN**, (defined in *<inttypes.h>*); resulting for example in **%"SCNd8"** or **%"SCNi8"** for scanning *int8_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *uintN_t*, and *uintmax_t* types in this page.

intptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

A signed integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [**INTPTR_MIN**, **INTPTR_MAX**].

The length modifier for *intptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIIdPTR** and **PRIiPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIIdPTR"** or **%"PRIiPTR"** for printing *intptr_t* values. The length modifier for *intptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNdPTR** and **SCNiPTR**, (defined in *<inttypes.h>*); resulting commonly in **%"SCNdPTR"** or **%"SCNiPTR"** for scanning *intptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *uintptr_t* and *void ** types in this page.

lconv

Include: *<locale.h>*.

```
struct lconv {                                /* Values in the "C" locale: */
    char    *decimal_point;                  /* ". " */
    char    *thousands_sep;                 /* " " */
    char    *grouping;                      /* " " */
    char    *mon_decimal_point;             /* " " */
    char    *mon_thousands_sep;             /* " " */
    char    *mon_grouping;                 /* " " */
    char    *positive_sign;                /* " " */
    char    *negative_sign;                /* " " */
    char    *currency_symbol;              /* " " */
    char    frac_digits;                   /* CHAR_MAX */
    char    p_cs_precedes;                /* CHAR_MAX */
    char    n_cs_precedes;                /* CHAR_MAX */
    char    p_sep_by_space;                /* CHAR_MAX */
    char    n_sep_by_space;                /* CHAR_MAX */
    char    p_sign_posn;                  /* CHAR_MAX */
    char    n_sign_posn;                  /* CHAR_MAX */
    char    *int_curr_symbol;              /* " " */
    char    int_frac_digits;               /* CHAR_MAX */
    char    int_p_cs_precedes;             /* CHAR_MAX */
    char    int_n_cs_precedes;             /* CHAR_MAX */
    char    int_p_sep_by_space;             /* CHAR_MAX */
    char    int_n_sep_by_space;             /* CHAR_MAX */
    char    int_p_sign_posn;               /* CHAR_MAX */
    char    int_n_sign_posn;               /* CHAR_MAX */
};
```

Contains members related to the formatting of numeric values. In the "C" locale, its members have the values shown in the comments above.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **setlocale**(3), **localeconv**(3), **charsets**(5), **locale**(7)

ldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long      quot; /* Quotient */
    long      rem;   /* Remainder */
} ldiv_t;
```

It is the type of the value returned by the **ldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **ldiv(3)**

lldiv_t

Include: <stdlib.h>.

```
typedef struct {
    long long  quot; /* Quotient */
    long long  rem;  /* Remainder */
} lldiv_t;
```

It is the type of the value returned by the **lldiv(3)** function.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **lldiv(3)**

off_t

Include: <sys/types.h>. Alternatively, <aio.h>, <fcntl.h>, <stdio.h>, <sys/mman.h>, <sys/stat.h>, or <unistd.h>.

Used for file sizes. According to POSIX, this shall be a signed integer type.

Versions: <aio.h> and <stdio.h> define *off_t* since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

Notes: On some architectures, the width of this type can be controlled with the feature test macro **_FILE_OFFSET_BITS**.

See also: **lseek(2)**, **mmap(2)**, **posix_fadvise(2)**, **pread(2)**, **truncate(2)**, **fseeko(3)**, **lockf(3)**, **posix_fallocate(3)**, **feature_test_macros(7)**

pid_t

Include: <sys/types.h>. Alternatively, <fcntl.h>, <sched.h>, <signal.h>, <spawn.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/wait.h>, <termios.h>, <time.h>, <unistd.h>, or <utmpx.h>.

This type is used for storing process IDs, process group IDs, and session IDs. According to POSIX, it shall be a signed integer type, and the implementation shall support one or more programming environments where the width of *pid_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: **fork(2)**, **getpid(2)**, **getppid(2)**, **getsid(2)**, **gettid(2)**, **getpgid(2)**, **kill(2)**, **pidfd_open(2)**, **sched_setscheduler(2)**, **waitpid(2)**, **sigqueue(3)**, **credentials(7)**,

ptrdiff_t

Include: <stddef.h>.

Used for a count of elements, and array indices. It is the result of subtracting two pointers. According to the C language standard, it shall be a signed integer type capable of storing values in the range [**PTRDIFF_MIN**, **PTRDIFF_MAX**].

The length modifier for *ptrdiff_t* for the **printf(3)** and the **scanf(3)** families of functions is **t**; resulting commonly in **%td** or **%ti** for printing *ptrdiff_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the `size_t` and `ssize_t` types in this page.

`regex_t`

Include: `<regex.h>`.

```
typedef struct {
    size_t re_nsub; /* Number of parenthesized subexpressions. */
} regex_t;
```

This is a structure type used in regular expression matching. It holds a compiled regular expression, compiled with `regcomp(3)`.

Conforming to: POSIX.1-2001 and later.

See also: `regex(3)`

`regmatch_t`

Include: `<regex.h>`.

```
typedef struct {
    regoff_t rm_so; /* Byte offset from start of string
                      to start of substring */
    regoff_t rm_eo; /* Byte offset from start of string of
                      the first character after the end of
                      substring */
} regmatch_t;
```

This is a structure type used in regular expression matching.

Conforming to: POSIX.1-2001 and later.

See also: `regexec(3)`

`regoff_t`

Include: `<regex.h>`.

According to POSIX, it shall be a signed integer type capable of storing the largest value that can be stored in either a `ptrdiff_t` type or a `ssize_t` type.

Versions: Prior to POSIX.1-2008, the type was capable of storing the largest value that can be stored in either an `off_t` type or a `ssize_t` type.

Conforming to: POSIX.1-2001 and later.

See also: the `regmatch_t` structure and the `ptrdiff_t` and `ssize_t` types in this page.

`sigevent`

Include: `<signal.h>`. Alternatively, `<aio.h>`, `<mqueue.h>`, or `<time.h>`.

```
struct sigevent {
    int           sigev_notify; /* Notification type */
    int           sigev_signo;  /* Signal number */
    union sigval  sigev_value; /* Signal value */
    void         (*sigev_notify_function)(union sigval);
                 /* Notification function */
    pthread_attr_t *sigev_notify_attributes;
                 /* Notification attributes */
};
```

For further details about this type, see `sigevent(7)`.

Versions: `<aio.h>` and `<time.h>` define `sigevent` since POSIX.1-2008.

Conforming to: POSIX.1-2001 and later.

See also: `timer_create(2)`, `getaddrinfo_a(3)`, `lio_listio(3)`, `mq_notify(3)`

See also the *aiocb* structure in this page.

siginfo_t

Include: <signal.h>. Alternatively, <sys/wait.h>.

```
typedef struct {
    int      si_signo; /* Signal number */
    int      si_code;  /* Signal code */
    pid_t   si_pid;   /* Sending process ID */
    uid_t   si_uid;   /* Real user ID of sending process */
    void    *si_addr; /* Address of faulting instruction */
    int     si_status; /* Exit value or signal */
    union sigval si_value; /* Signal value */
} siginfo_t;
```

Information associated with a signal. For further details on this structure (including additional, Linux-specific fields), see **sigaction**(2).

Conforming to: POSIX.1-2001 and later.

See also: **pidfd_send_signal**(2), **rt_sigqueueinfo**(2), **sigaction**(2), **sigwaitinfo**(2), **psiginfo**(3)

sigset_t

Include: <signal.h>. Alternatively, <spawn.h>, or <sys/select.h>.

This is a type that represents a set of signals. According to POSIX, this shall be an integer or structure type.

Conforming to: POSIX.1-2001 and later.

See also: **epoll_pwait**(2), **ppoll**(2), **pselect**(2), **sigaction**(2), **signalfd**(2), **sigpending**(2), **sigprocmask**(2), **sigsuspend**(2), **sigwaitinfo**(2), **signal**(7)

sigval

Include: <signal.h>.

```
union sigval {
    int     sigval_int; /* Integer value */
    void   *sigval_ptr; /* Pointer value */
};
```

Data passed with a signal.

Conforming to: POSIX.1-2001 and later.

See also: **pthread_sigqueue**(3), **sigqueue**(3), **sigevent**(7)

See also the *sigevent* structure and the *siginfo_t* type in this page.

size_t

Include: <stddef.h> or <sys/types.h>. Alternatively, <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <monetary.h>, <mqueue.h>, <ndbm.h>, <pwd.h>, <regex.h>, <search.h>, <signal.h>, <stdio.h>, <stdlib.h>, <string.h>, <strings.h>, <sys/mman.h>, <sys/msg.h>, <sys/sem.h>, <sys/shm.h>, <sys/socket.h>, <sys/uio.h>, <time.h>, <unistd.h>, <wchar.h>, or <wctype.h>.

Used for a count of bytes. It is the result of the *sizeof* operator. According to the C language standard, it shall be an unsigned integer type capable of storing values in the range [0, **SIZE_MAX**]. According to POSIX, the implementation shall support one or more programming environments where the width of *size_t* is no greater than the width of the type *long*.

The length modifier for *size_t* for the **printf**(3) and the **scanf**(3) families of functions is **z**; resulting commonly in **%zu** or **%zx** for printing *size_t* values.

Versions: <aio.h>, <glob.h>, <grp.h>, <iconv.h>, <mqueue.h>, <pwd.h>, <signal.h>, and <sys/socket.h> define *size_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **read(2)**, **write(2)**, **fread(3)**, **fwrite(3)**, **memcmp(3)**, **memcpy(3)**, **memset(3)**, **off-setof(3)**

See also the *ptrdiff_t* and *ssize_t* types in this page.

ssize_t

Include: <sys/types.h>. Alternatively, <aio.h>, <monetary.h>, <mqueue.h>, <stdio.h>, <sys/msg.h>, <sys/socket.h>, <sys/uio.h>, or <unistd.h>.

Used for a count of bytes or an error indication. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, **SSIZE_MAX**], and the implementation shall support one or more programming environments where the width of *ssize_t* is no greater than the width of the type *long*.

Glibc and most other implementations provide a length modifier for *ssize_t* for the **printf(3)** and the **scanf(3)** families of functions, which is **z**; resulting commonly in **%zd** or **%zi** for printing *ssize_t* values. Although **z** works for *ssize_t* on most implementations, portable POSIX programs should avoid using it—for example, by converting the value to *intmax_t* and using its length modifier (**j**).

Conforming to: POSIX.1-2001 and later.

See also: **read(2)**, **readlink(2)**, **readv(2)**, **recv(2)**, **send(2)**, **write(2)**

See also the *ptrdiff_t* and *size_t* types in this page.

suseconds_t

Include: <sys/types.h>. Alternatively, <sys/select.h>, or <sys/time.h>.

Used for time in microseconds. According to POSIX, it shall be a signed integer type capable of storing values at least in the range [-1, 1000000], and the implementation shall support one or more programming environments where the width of *suseconds_t* is no greater than the width of the type *long*.

Conforming to: POSIX.1-2001 and later.

See also: the *timeval* structure in this page.

time_t

Include: <time.h> or <sys/types.h>. Alternatively, <sched.h>, <sys/msg.h>, <sys/select.h>, <sys/sem.h>, <sys/shm.h>, <sys/stat.h>, <sys/time.h>, or <utime.h>.

Used for time in seconds. According to POSIX, it shall be an integer type.

Versions: <sched.h> defines *time_t* since POSIX.1-2008.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **stime(2)**, **time(2)**, **ctime(3)**, **difftime(3)**

timer_t

Include: <sys/types.h>. Alternatively, <time.h>.

Used for timer ID returned by **timer_create(2)**. According to POSIX, there are no defined comparison or assignment operators for this type.

Conforming to: POSIX.1-2001 and later.

See also: **timer_create(2)**, **timer_delete(2)**, **timer_getoverrun(2)**, **timer_settime(2)**

timespec

Include: <time.h>. Alternatively, <aio.h>, <mqueue.h>, <sched.h>, <signal.h>, <sys/select.h>, or <sys/stat.h>.

```
struct timespec {
    time_t tv_sec; /* Seconds */
```

```
    long      tv_nsec; /* Nanoseconds */
}
```

Describes times in seconds and nanoseconds.

Conforming to: C11 and later; POSIX.1-2001 and later.

See also: **clock_gettime(2)**, **clock_nanosleep(2)**, **nanosleep(2)**, **timerfd_gettime(2)**, **timer_gettime(2)**

timeval

Include: <sys/time.h>. Alternatively, <sys/resource.h>, <sys/select.h>, or <utmpx.h>.

```
struct timeval {
    time_t      tv_sec;   /* Seconds */
    suseconds_t tv_usec; /* Microseconds */
};
```

Describes times in seconds and microseconds.

Conforming to: POSIX.1-2001 and later.

See also: **gettimeofday(2)**, **select(2)**, **utimes(2)**, **adjtime(3)**, **futimes(3)**, **timeradd(3)**

uid_t

Include: <sys/types.h>. Alternatively, <pwd.h>, <signal.h>, <stropts.h>, <sys/ipc.h>, <sys/stat.h>, or <unistd.h>.

A type used to hold user IDs. According to POSIX, this shall be an integer type.

Conforming to: POSIX.1-2001 and later.

See also: **chown(2)**, **getuid(2)**, **geteuid(2)**, **getresuid(2)**, **getpwnam(2)**, **credentials(7)**

uintmax_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

An unsigned integer type capable of representing any value of any unsigned integer type supported by the implementation. According to the C language standard, it shall be capable of storing values in the range [0, **UINTMAX_MAX**].

The macro **UINTMAX_C()** expands its argument to an integer constant of type *uintmax_t*.

The length modifier for *uintmax_t* for the **printf(3)** and the **scanf(3)** families of functions is **j**; resulting commonly in **%ju** or **%jx** for printing *uintmax_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

Bugs: *uintmax_t* is not large enough to represent values of type *unsigned __int128* in implementations where *unsigned __int128* is defined and *unsigned long long* is less than 128 bits wide.

See also: the *intmax_t* type in this page.

uintN_t

Include: <stdint.h>. Alternatively, <inttypes.h>.

uint8_t, *uint16_t*, *uint32_t*, *uint64_t*

An unsigned integer type of a fixed width of exactly N bits, N being the value specified in its type name. According to the C language standard, they shall be capable of storing values in the range [0, **UINTN_MAX**], substituting N by the appropriate number.

According to POSIX, *uint8_t*, *uint16_t*, and *uint32_t* are required; *uint64_t* is only required in implementations that provide integer types with width 64; and all other types of this form are optional.

The length modifiers for the *uintN_t* types for the **printf(3)** family of functions are expanded by macros of the forms **PRIuN**, **PRIoN**, **PRIxN**, and **PRIXN** (defined in <inttypes.h>); resulting for example in **%"PRIu32"** or **%"PRIx32"** for printing *uint32_t* values. The length modifiers for

the *uintN_t* types for the **scanf**(3) family of functions are expanded by macros of the forms **SCNuN**, **SCNoN**, **SCNxN**, and **SCNXN** (defined in *<inttypes.h>*); resulting for example in **%"SCNu16"** or **%"SCNx16"** for scanning *uint16_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intmax_t*, *intN_t*, and *uintmax_t* types in this page.

uintptr_t

Include: *<stdint.h>*. Alternatively, *<inttypes.h>*.

An unsigned integer type such that any valid (*void **) value can be converted to this type and back. According to the C language standard, it shall be capable of storing values in the range [0, **UINTPTR_MAX**].

The length modifier for *uintptr_t* for the **printf**(3) family of functions is expanded by the macros **PRIupTR**, **PRIoPTR**, **PRIxPTR**, and **PRIxPTR** (defined in *<inttypes.h>*); resulting commonly in **%"PRIupTR"** or **%"PRIxPTR"** for printing *uintptr_t* values. The length modifier for *uintptr_t* for the **scanf**(3) family of functions is expanded by the macros **SCNuPTR**, **SCNoPTR**, **SCNxPTR**, and **SCNXPTR** (defined in *<inttypes.h>*); resulting commonly in **%"SCNuPTR"** or **%"SCNxPTR"** for scanning *uintptr_t* values.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: the *intptr_t* and *void ** types in this page.

va_list

Include: *<stdarg.h>*. Alternatively, *<stdio.h>*, or *<wchar.h>*.

Used by functions with a varying number of arguments of varying types. The function must declare an object of type *va_list* which is used by the macros **va_start**(3), **va_arg**(3), **va_copy**(3), and **va_end**(3) to traverse the list of arguments.

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **va_start**(3), **va_arg**(3), **va_copy**(3), **va_end**(3)

*void **

According to the C language standard, a pointer to any object type may be converted to a pointer to *void* and back. POSIX further requires that any pointer, including pointers to functions, may be converted to a pointer to *void* and back.

Conversions from and to any other pointer type are done implicitly, not requiring casts at all. Note that this feature prevents any kind of type checking: the programmer should be careful not to convert a *void ** value to a type incompatible to that of the underlying data, because that would result in undefined behavior.

This type is useful in function parameters and return value to allow passing values of any type. The function will typically use some mechanism to know the real type of the data being passed via a pointer to *void*.

A value of this type can't be dereferenced, as it would give a value of type *void*, which is not possible. Likewise, pointer arithmetic is not possible with this type. However, in GNU C, pointer arithmetic is allowed as an extension to the standard; this is done by treating the size of a *void* or of a function as 1. A consequence of this is that *sizeof* is also allowed on *void* and on function types, and returns 1.

The conversion specifier for *void ** for the **printf**(3) and the **scanf**(3) families of functions is **p**.

Versions: The POSIX requirement about compatibility between *void ** and function pointers was added in POSIX.1-2008 Technical Corrigendum 1 (2013).

Conforming to: C99 and later; POSIX.1-2001 and later.

See also: **malloc**(3), **memcmp**(3), **memcpy**(3), **memset**(3)

See also the *intptr_t* and *uintptr_t* types in this page.

NOTES

The structures described in this manual page shall contain, at least, the members shown in their definition, in no particular order.

Most of the integer types described in this page don't have a corresponding length modifier for the `printf(3)` and the `scanf(3)` families of functions. To print a value of an integer type that doesn't have a length modifier, it should be converted to `intmax_t` or `uintmax_t` by an explicit cast. To scan into a variable of an integer type that doesn't have a length modifier, an intermediate temporary variable of type `intmax_t` or `uintmax_t` should be used. When copying from the temporary variable to the destination variable, the value could overflow. If the type has upper and lower limits, the user should check that the value is within those limits, before actually copying the value. The example below shows how these conversions should be done.

Conventions used in this page

In "Conforming to" we only concern ourselves with C99 and later and POSIX.1-2001 and later. Some types may be specified in earlier versions of one of these standards, but in the interests of simplicity we omit details from earlier standards.

In "Include", we first note the "primary" header(s) that define the type according to either the C or POSIX.1 standards. Under "Alternatively", we note additional headers that the standards specify shall define the type.

EXAMPLES

The program shown below scans from a string and prints a value stored in a variable of an integer type that doesn't have a length modifier. The appropriate conversions from and to `intmax_t`, and the appropriate range checks, are used as explained in the notes section above.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int
main (void)
{
    static const char *const str = "500000 us in half a second";
    suseconds_t us;
    intmax_t tmp;

    /* Scan the number from the string into the temporary variable */

    sscanf(str, "%jd", &tmp);

    /* Check that the value is within the valid range of suseconds_t */

    if (tmp < -1 || tmp > 1000000) {
        fprintf(stderr, "Scanned value outside valid range!\n");
        exit(EXIT_FAILURE);
    }

    /* Copy the value to the suseconds_t variable 'us' */

    us = tmp;

    /* Even though suseconds_t can hold the value -1, this isn't
       a sensible number of microseconds */
}
```

```
if (us < 0) {
    fprintf(stderr, "Scanned value shouldn't be negative!\n");
    exit(EXIT_FAILURE);
}

/* Print the value */

printf("There are %jd microseconds in half a second.\n",
       (intmax_t) us);

exit(EXIT_SUCCESS);
}
```

SEE ALSO

[feature_test_macros\(7\)](#), [standards\(7\)](#)

COLPHON

This page is part of release 5.10 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.

NAME

timedatectl – Control the system time and date

SYNOPSIS

timedatectl [OPTIONS...] {COMMAND}

DESCRIPTION

timedatectl may be used to query and change the system clock and its settings, and enable or disable time synchronization services.

Use **systemd-firstboot(1)** to initialize the system time zone for mounted (but not booted) system images.

timedatectl may be used to show the current status of time synchronization services, for example **systemd-timesyncd.service(8)**.

COMMANDS

The following commands are understood:

status

Show current settings of the system clock and RTC, including whether network time synchronization is active. If no command is specified, this is the implied default.

show

Show the same information as **status**, but in machine readable form. This command is intended to be used whenever computer-parsable output is required. Use **status** if you are looking for formatted human-readable output.

By default, empty properties are suppressed. Use **--all** to show those too. To select specific properties to show, use **--property=**.

set-time [TIME]

Set the system clock to the specified time. This will also update the RTC time accordingly. The time may be specified in the format "2012-10-30 18:17:16".

set-timezone [TIMEZONE]

Set the system time zone to the specified value. Available timezones can be listed with **list-timezones**. If the RTC is configured to be in the local time, this will also update the RTC time. This call will alter the /etc/localtime symlink. See **localtime(5)** for more information.

list-timezones

List available time zones, one per line. Entries from the list can be set as the system timezone with **set-timezone**.

set-local-rtc [BOOL]

Takes a boolean argument. If "0", the system is configured to maintain the RTC in universal time. If "1", it will maintain the RTC in local time instead. Note that maintaining the RTC in the local timezone is not fully supported and will create various problems with time zone changes and daylight saving adjustments. If at all possible, keep the RTC in UTC mode. Note that invoking this will also synchronize the RTC from the system clock, unless **--adjust-system-clock** is passed (see above). This command will change the 3rd line of /etc/adjtime, as documented in **hwclock(8)**.

set-ntp [BOOL]

Takes a boolean argument. Controls whether network time synchronization is active and enabled (if available). If the argument is true, this enables and starts the first existing network synchronization service. If the argument is false, then this disables and stops the known network synchronization services. The way that the list of services is built is described in **systemd-timedated.service(8)**.

systemd-timesyncd Commands

The following commands are specific to **systemd-timesyncd.service(8)**.

timesync-status

Show current status of **systemd-timesyncd.service(8)**. If **--monitor** is specified, then this will monitor the status updates.

show-timesync

Show the same information as **timesync-status**, but in machine readable form. This command is intended to be used whenever computer-parsable output is required. Use **timesync-status** if you are looking for formatted human-readable output.

By default, empty properties are suppressed. Use **--all** to show those too. To select specific properties to show, use **--property=**.

ntp-servers INTERFACE SERVER...

Set the interface specific NTP servers. This command can be used only when the interface is managed by **systemd-networkd**.

revert INTERFACE

Revert the interface specific NTP servers. This command can be used only when the interface is managed by **systemd-networkd**.

OPTIONS

The following options are understood:

--no-ask-password

Do not query the user for authentication for privileged operations.

--adjust-system-clock

If **set-local-rtc** is invoked and this option is passed, the system clock is synchronized from the RTC again, taking the new setting into account. Otherwise, the RTC is synchronized from the system clock.

--monitor

If **timesync-status** is invoked and this option is passed, then **timedatectl** monitors the status of **systemd-timesyncd.service(8)** and updates the outputs. Use Ctrl+C to terminate the monitoring.

-a, --all

When showing properties of **systemd-timesyncd.service(8)**, show all properties regardless of whether they are set or not.

-p, --property=

When showing properties of **systemd-timesyncd.service(8)**, limit display to certain properties as specified as argument. If not specified, all set properties are shown. The argument should be a property name, such as "ServerName". If specified more than once, all properties with the specified names are shown.

--value

When printing properties with **show-timesync**, only print the value, and skip the property name and "**=**".

-H, --host=

Execute the operation remotely. Specify a hostname, or a username and hostname separated by "@", to connect to. The hostname may optionally be suffixed by a port ssh is listening on, separated by ":"; and then a container name, separated by "/", which connects directly to a specific container on the specified host. This will use SSH to talk to the remote machine manager instance. Container names may be enumerated with **machinectl -H HOST**. Put IPv6 addresses in brackets.

-M, --machine=

Execute operation on a local container. Specify a container name to connect to, optionally prefixed by a user name to connect as and a separating "@" character. If the special string ".host" is used in place of the container name, a connection to the local system is made (which is useful to connect to a specific user's user bus: "--user --machine=lennart@.host"). If the "@" syntax is not used, the connection is made as root user. If the "@" syntax is used either the left hand side or the right hand side may be omitted (but not both) in which case the local user name and ".host" are implied.

-h, --help

Print a short help text and exit.

--version

Print a short version string and exit.

--no-pager

Do not pipe output into a pager.

EXIT STATUS

On success, 0 is returned, a non-zero failure code otherwise.

ENVIRONMENT***\$SYSTEMD_LOG_LEVEL***

The maximum log level of emitted messages (messages with a higher log level, i.e. less important ones, will be suppressed). Either one of (in order of decreasing importance) **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, **debug**, or an integer in the range 0...7. See **syslog(3)** for more information.

\$SYSTEMD_LOG_COLOR

A boolean. If true, messages written to the tty will be colored according to priority.

This setting is only useful when messages are written directly to the terminal, because **journalctl(1)** and other tools that display logs will color messages based on the log level on their own.

\$SYSTEMD_LOG_TIME

A boolean. If true, console log messages will be prefixed with a timestamp.

This setting is only useful when messages are written directly to the terminal or a file, because **journalctl(1)** and other tools that display logs will attach timestamps based on the entry metadata on their own.

\$SYSTEMD_LOG_LOCATION

A boolean. If true, messages will be prefixed with a filename and line number in the source code where the message originates.

Note that the log location is often attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TID

A boolean. If true, messages will be prefixed with the current numerical thread ID (TID).

Note that this information is attached as metadata to journal entries anyway. Including it directly in the message text can nevertheless be convenient when debugging programs.

\$SYSTEMD_LOG_TARGET

The destination for log messages. One of **console** (log to the attached tty), **console-prefixed** (log to the attached tty but with prefixes encoding the log level and "facility", see **syslog(3)**), **kmsg** (log to the kernel circular log buffer), **journal** (log to the journal), **journal-or-kmsg** (log to the journal if available, and to kmsg otherwise), **auto** (determine the appropriate log target automatically, the default), **null** (disable log output).

\$SYSTEMD_PAGER

Pager to use when **--no-pager** is not given; overrides **\$PAGER**. If neither **\$SYSTEMD_PAGER** nor **\$PAGER** are set, a set of well-known pager implementations are tried in turn, including **less(1)** and **more(1)**, until one is found. If no pager implementation is discovered no pager is invoked. Setting this environment variable to an empty string or the value "cat" is equivalent to passing **--no-pager**.

\$SYSTEMD_LESS

Override the options passed to **less** (by default "FRSXMK").

Users might want to change two options in particular:

K

This option instructs the pager to exit immediately when Ctrl+C is pressed. To allow **less** to

handle Ctrl+C itself to switch back to the pager command prompt, unset this option.

If the value of `$SYSTEMD_LESS` does not include "K", and the pager that is invoked is `less`, Ctrl+C will be ignored by the executable, and needs to be handled by the pager.

X

This option instructs the pager to not send termcap initialization and deinitialization strings to the terminal. It is set by default to allow command output to remain visible in the terminal even after the pager exits. Nevertheless, this prevents some pager functionality from working, in particular paged output cannot be scrolled with the mouse.

See `less(1)` for more discussion.

`$SYSTEMD_LESSCHARSET`

Override the charset passed to `less` (by default "utf-8", if the invoking terminal is determined to be UTF-8 compatible).

`$SYSTEMD_PAGERSECURE`

Takes a boolean argument. When true, the "secure" mode of the pager is enabled; if false, disabled. If `$SYSTEMD_PAGERSECURE` is not set at all, secure mode is enabled if the effective UID is not the same as the owner of the login session, see `geteuid(2)` and `sd_pid_get_owner_uid(3)`. In secure mode, `LESSSECURE=1` will be set when invoking the pager, and the pager shall disable commands that open or create new files or start new subprocesses. When `$SYSTEMD_PAGERSECURE` is not set at all, pagers which are not known to implement secure mode will not be used. (Currently only `less(1)` implements secure mode.)

Note: when commands are invoked with elevated privileges, for example under `sudo(8)` or `pkexec(1)`, care must be taken to ensure that unintended interactive features are not enabled. "Secure" mode for the pager may be enabled automatically as described above. Setting `SYSTEMD_PAGERSECURE=0` or not removing it from the inherited environment allows the user to invoke arbitrary commands. Note that if the `$SYSTEMD_PAGER` or `$PAGER` variables are to be honoured, `$SYSTEMD_PAGERSECURE` must be set too. It might be reasonable to completely disable the pager using `--no-pager` instead.

`$SYSTEMD_COLORS`

Takes a boolean argument. When true, `systemd` and related utilities will use colors in their output, otherwise the output will be monochrome. Additionally, the variable can take one of the following special values: "16", "256" to restrict the use of colors to the base 16 or 256 ANSI colors, respectively. This can be specified to override the automatic decision based on `$TERM` and what the console is connected to.

`$SYSTEMD_URLIFY`

The value must be a boolean. Controls whether clickable links should be generated in the output for terminal emulators supporting this. This can be specified to override the decision that `systemd` makes based on `$TERM` and other conditions.

EXAMPLES

Show current settings:

```
$ timedatectl
  Local time: Thu 2017-09-21 16:08:56 CEST
  Universal time: Thu 2017-09-21 14:08:56 UTC
    RTC time: Thu 2017-09-21 14:08:56
    Time zone: Europe/Warsaw (CEST, +0200)
System clock synchronized: yes
      NTP service: active
      RTC in local TZ: no
```

Enable network time synchronization:

```
$ timedatectl set-ntp true
===== AUTHENTICATING FOR org.freedesktop.timedate1.set-ntp ====
Authentication is required to control whether network time synchronization shall be enabled.
Authenticating as: user
Password: *****
===== AUTHENTICATION COMPLETE ===
```

```
$ systemctl status systemd-timesyncd.service
systemd-timesyncd.service – Network Time Synchronization
   Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled)
   Active: active (running) since Mo 2015-03-30 14:20:38 CEST; 5s ago
     Docs: man:systemd-timesyncd.service(8)
 Main PID: 595 (systemd-timesyn)
    Status: "Using Time Server 216.239.38.15:123 (time4.google.com)."
   CGroup: /system.slice/systemd-timesyncd.service
          595 /lib/systemd/systemd-timesyncd
...

```

Show current status of **systemd-timesyncd.service(8)**:

```
$ timedatectl timesync-status
  Server: 216.239.38.15 (time4.google.com)
  Poll interval: 1min 4s (min: 32s; max 34min 8s)
    Leap: normal
    Version: 4
    Stratum: 1
    Reference: GPS
    Precision: 1us (-20)
  Root distance: 335us (max: 5s)
    Offset: +316us
    Delay: 349us
    Jitter: 0
  Packet count: 1
    Frequency: -8.802ppm
```

SEE ALSO

systemd(1), hwclock(8), date(1), localtime(5), systemctl(1), systemd-timedated.service(8), systemd-timesyncd.service(8), systemd-firstboot(1)

NAME

times – get process times

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/times.h>
clock_t times(struct tms *buf);
```

DESCRIPTION

times() stores the current process times in the *struct tms* that *buf* points to. The *struct tms* is as defined in *<sys/times.h>*:

```
struct tms {
    clock_t tms_utime; /* user time */
    clock_t tms_stime; /* system time */
    clock_t tms_cutime; /* user time of children */
    clock_t tms_cstime; /* system time of children */
};
```

The *tms_utime* field contains the CPU time spent executing instructions of the calling process. The *tms_stime* field contains the CPU time spent executing inside the kernel while performing tasks on behalf of the calling process.

The *tms_cutime* field contains the sum of the *tms_utime* and *tms_cutime* values for all waited-for terminated children. The *tms_cstime* field contains the sum of the *tms_stime* and *tms_cstime* values for all waited-for terminated children.

Times for terminated children (and their descendants) are added in at the moment **wait(2)** or **waitpid(2)** returns their process ID. In particular, times of grandchildren that the children did not wait for are never seen.

All times reported are in clock ticks.

RETURN VALUE

times() returns the number of clock ticks that have elapsed since an arbitrary point in the past. The return value may overflow the possible range of type *clock_t*. On error, *(clock_t) -1* is returned, and *errno* is set to indicate the error.

ERRORS**EFAULT**

tms points outside the process's address space.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD.

NOTES

The number of clock ticks per second can be obtained using:

```
sysconf(_SC_CLK_TCK);
```

In POSIX.1-1996 the symbol **CLK_TCK** (defined in *<time.h>*) is mentioned as obsolescent. It is obsolete now.

Before Linux 2.6.9, if the disposition of **SIGCHLD** is set to **SIG_IGN**, then the times of terminated children are automatically included in the *tms_cstime* and *tms_cutime* fields, although POSIX.1-2001 says that this should happen only if the calling process **wait(2)**s on its children. This nonconformance is rectified in Linux 2.6.9 and later.

On Linux, the *buf* argument can be specified as NULL, with the result that **times()** just returns a function result. However, POSIX does not specify this behavior, and most other UNIX implementations require a non-NULL value for *buf*.

Note that **clock(3)** also returns a value of type *clock_t*, but this value is measured in units of

CLOCKS_PER_SEC, not the clock ticks used by **times()**.

On Linux, the “arbitrary point in the past” from which the return value of **times()** is measured has varied across kernel versions. On Linux 2.4 and earlier, this point is the moment the system was booted. Since Linux 2.6, this point is $(2^{32}/HZ) - 300$ seconds before system boot time. This variability across kernel versions (and across UNIX implementations), combined with the fact that the returned value may overflow the range of *clock_t*, means that a portable application would be wise to avoid using this value. To measure changes in elapsed time, use **clock_gettime(2)** instead.

Historical

SVr1-3 returns *long* and the struct members are of type *time_t* although they store clock ticks, not seconds since the Epoch. V7 used *long* for the struct members, because it had no type *time_t* yet.

BUGS

A limitation of the Linux system call conventions on some architectures (notably i386) means that on Linux 2.6 there is a small time window (41 seconds) soon after boot when **times()** can return -1, falsely indicating that an error occurred. The same problem can occur when the return value wraps past the maximum value that can be stored in **clock_t**.

SEE ALSO

time(1), **getrusage(2)**, **wait(2)**, **clock(3)**, **sysconf(3)**, **time(7)**

NAME

timesyncd.conf, timesyncd.conf.d – Network Time Synchronization configuration files

SYNOPSIS

```
/etc/systemd/timesyncd.conf
/etc/systemd/timesyncd.conf.d/*.conf
/run/systemd/timesyncd.conf.d/*.conf
/usr/lib/systemd/timesyncd.conf.d/*.conf
```

DESCRIPTION

These configuration files control NTP network time synchronization. See **systemd.syntax(7)** for a general description of the syntax.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in /etc/systemd/ contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from /usr/lib/systemd/*.conf.d/, /usr/local/lib/systemd/*.conf.d/, and /etc/systemd/*.conf.d/. Those drop-ins have higher precedence and override the main configuration file. Files in the *.conf.d/ configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under /usr/. Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file.

OPTIONS

The following settings are configured in the [Time] section:

NTP=

A space-separated list of NTP server host names or IP addresses. During runtime this list is combined with any per-interface NTP servers acquired from **systemd-networkd.service(8)**.

systemd-timesyncd will contact all configured system or per-interface servers in turn, until one responds. When the empty string is assigned, the list of NTP servers is reset, and all prior assignments will have no effect. This setting defaults to an empty list.

FallbackNTP=

A space-separated list of NTP server host names or IP addresses to be used as the fallback NTP servers. Any per-interface NTP servers obtained from **systemd-networkd.service(8)** take precedence over this setting, as do any servers set via *NTP=* above. This setting is hence only relevant if no other NTP server information is known. When the empty string is assigned, the list of NTP servers is reset, and all prior assignments will have no effect. If this option is not given, a compiled-in list of NTP servers is used.

RootDistanceMaxSec=

Maximum acceptable root distance, i.e. the maximum estimated time required for a packet to travel to the server we are connected to from the server with the reference clock. If the current server does not satisfy this limit, **systemd-timesyncd** will switch to a different server.

Takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. Defaults to 5 seconds.

PollIntervalMinSec=, *PollIntervalMaxSec=*

The minimum and maximum poll intervals for NTP messages. Polling starts at the minimum poll interval, and is adjusted within the specified limits in response to received packets.

Each setting takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. *PollIntervalMinSec=* defaults to 32 seconds and must not be smaller than 16 seconds. *PollIntervalMaxSec=* defaults to 34 min 8 s (2048 seconds) and must be larger than *PollIntervalMinSec=*.

ConnectionRetrySec=

Specifies the minimum delay before subsequent attempts to contact a new NTP server are made.

Takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. Defaults to 30 seconds and must not be smaller than 1 second.

SEE ALSO

systemd(1), **systemd-timesyncd.service(8)**, **systemd-networkd.service(8)**

NAME

timesyncd.conf, timesyncd.conf.d – Network Time Synchronization configuration files

SYNOPSIS

```
/etc/systemd/timesyncd.conf
/etc/systemd/timesyncd.conf.d/*.conf
/run/systemd/timesyncd.conf.d/*.conf
/usr/lib/systemd/timesyncd.conf.d/*.conf
```

DESCRIPTION

These configuration files control NTP network time synchronization. See **systemd.syntax(7)** for a general description of the syntax.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is set during compilation, so configuration is only needed when it is necessary to deviate from those defaults. Initially, the main configuration file in /etc/systemd/ contains commented out entries showing the defaults as a guide to the administrator. Local overrides can be created by editing this file or by creating drop-ins, as described below. Using drop-ins for local configuration is recommended over modifications to the main configuration file.

In addition to the "main" configuration file, drop-in configuration snippets are read from /usr/lib/systemd/*.conf.d/, /usr/local/lib/systemd/*.conf.d/, and /etc/systemd/*.conf.d/. Those drop-ins have higher precedence and override the main configuration file. Files in the *.conf.d/ configuration subdirectories are sorted by their filename in lexicographic order, regardless of in which of the subdirectories they reside. When multiple files specify the same option, for options which accept just a single value, the entry in the file sorted last takes precedence, and for options which accept a list of values, entries are collected as they occur in the sorted files.

When packages need to customize the configuration, they can install drop-ins under /usr/. Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. Drop-ins have to be used to override package drop-ins, since the main configuration file has lower precedence. It is recommended to prefix all filenames in those subdirectories with a two-digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file.

OPTIONS

The following settings are configured in the [Time] section:

NTP=

A space-separated list of NTP server host names or IP addresses. During runtime this list is combined with any per-interface NTP servers acquired from **systemd-networkd.service(8)**.

systemd-timesyncd will contact all configured system or per-interface servers in turn, until one responds. When the empty string is assigned, the list of NTP servers is reset, and all prior assignments will have no effect. This setting defaults to an empty list.

FallbackNTP=

A space-separated list of NTP server host names or IP addresses to be used as the fallback NTP servers. Any per-interface NTP servers obtained from **systemd-networkd.service(8)** take precedence over this setting, as do any servers set via *NTP=* above. This setting is hence only relevant if no other NTP server information is known. When the empty string is assigned, the list of NTP servers is reset, and all prior assignments will have no effect. If this option is not given, a compiled-in list of NTP servers is used.

RootDistanceMaxSec=

Maximum acceptable root distance, i.e. the maximum estimated time required for a packet to travel to the server we are connected to from the server with the reference clock. If the current server does not satisfy this limit, **systemd-timesyncd** will switch to a different server.

Takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. Defaults to 5 seconds.

PollIntervalMinSec=, *PollIntervalMaxSec=*

The minimum and maximum poll intervals for NTP messages. Polling starts at the minimum poll interval, and is adjusted within the specified limits in response to received packets.

Each setting takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. *PollIntervalMinSec=* defaults to 32 seconds and must not be smaller than 16 seconds. *PollIntervalMaxSec=* defaults to 34 min 8 s (2048 seconds) and must be larger than *PollIntervalMinSec=*.

ConnectionRetrySec=

Specifies the minimum delay before subsequent attempts to contact a new NTP server are made.

Takes a time span value. The default unit is seconds, but other units may be specified, see **systemd.time(5)**. Defaults to 30 seconds and must not be smaller than 1 second.

SEE ALSO

systemd(1), **systemd-timesyncd.service(8)**, **systemd-networkd.service(8)**

NAME

tkill, **tkill** – send a signal to a thread

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <signal.h>      /* Definition of SIG* constants */
#include <sys/syscall.h>   /* Definition of SYS_* constants */
#include <unistd.h>

[[deprecated]] int syscall(SYS_tkill, pid_t tid, int sig);

#include <signal.h>

int tkill(pid_t tgid, pid_t tid, int sig);
```

Note: glibc provides no wrapper for **tkill()**, necessitating the use of **syscall(2)**.

DESCRIPTION

tgkill() sends the signal *sig* to the thread with the thread ID *tid* in the thread group *tgid*. (By contrast, **kill(2)** can be used to send a signal only to a process (i.e., thread group) as a whole, and the signal will be delivered to an arbitrary thread within that process.)

tkill() is an obsolete predecessor to **tgkill()**. It allows only the target thread ID to be specified, which may result in the wrong thread being signaled if a thread terminates and its thread ID is recycled. Avoid using this system call.

These are the raw system call interfaces, meant for internal thread library use.

RETURN VALUE

On success, zero is returned. On error, **-1** is returned, and *errno* is set to indicate the error.

ERRORS

EAGAIN

The **RLIMIT_SIGPENDING** resource limit was reached and *sig* is a real-time signal.

EAGAIN

Insufficient kernel memory was available and *sig* is a real-time signal.

EINVAL

An invalid thread ID, thread group ID, or signal was specified.

EPERM

Permission denied. For the required permissions, see **kill(2)**.

ESRCH

No process with the specified thread ID (and thread group ID) exists.

VERSIONS

tkill() is supported since Linux 2.4.19 / 2.5.4. **tgkill()** was added in Linux 2.5.75.

Library support for **tgkill()** was added in glibc 2.30.

STANDARDS

tkill() and **tgkill()** are Linux-specific and should not be used in programs that are intended to be portable.

NOTES

See the description of **CLONE_THREAD** in **clone(2)** for an explanation of thread groups.

Before glibc 2.30, there was also no wrapper function for **tgkill()**.

SEE ALSO

clone(2), **gettid(2)**, **kill(2)**, **rt_sigqueueinfo(2)**

NAME

top – display Linux processes

SYNOPSIS

```
top [-hv|bcEeHiOSs1] [-d secs] [-n max] [-u|U user] [-p pids] [-o field] [-w [cols]]
```

The traditional switches ‘–’ and whitespace are optional.

DESCRIPTION

The **top** program provides a dynamic real-time view of a running system. It can display **system** summary information as well as a list of **processes** or **threads** currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts.

The program provides a limited interactive interface for process manipulation as well as a much more extensive interface for personal configuration — encompassing every aspect of its operation. And while **top** is referred to throughout this document, you are free to name the program anything you wish. That new name, possibly an alias, will then be reflected on top’s display and used when reading and writing a configuration file.

OVERVIEW**Documentation**

The remaining Table of Contents

- OVERVIEW
 - Operation
 - Linux Memory Types
 - 1. COMMAND-LINE Options
 - 2. SUMMARY Display
 - a. UPTIME and LOAD Averages
 - b. TASK and CPU States
 - c. MEMORY Usage
 - 3. FIELDS / Columns Display
 - a. DESCRIPTIONS of Fields
 - b. MANAGING Fields
 - 4. INTERACTIVE Commands
 - a. GLOBAL Commands
 - b. SUMMARY AREA Commands
 - c. TASK AREA Commands
 - 1. Appearance
 - 2. Content
 - 3. Size
 - 4. Sorting
 - d. COLOR Mapping
 - 5. ALTERNATE-DISPLAY Provisions
 - a. WINDOWS Overview
 - b. COMMANDS for Windows
 - c. SCROLLING a Window
 - d. SEARCHING in a Window
 - e. FILTERING in a Window
 - 6. FILES
 - a. PERSONAL Configuration File
 - b. ADDING INSPECT Entries

- c. SYSTEM Configuration File
- d. SYSTEM Restrictions File
- 7. STUPID TRICKS Sampler
 - a. Kernel Magic
 - b. Bouncing Windows
 - c. The Big Bird Window
 - d. The Ol' Switcheroo
- 8. BUGS, 9. SEE Also

Operation

When operating top, the two most important keys are the help (h or ?) key and quit ('q') key. Alternatively, you could simply use the traditional interrupt key (^C) when you're done.

When started for the first time, you'll be presented with these traditional elements on the main top screen: 1) Summary Area; 2) Fields/Columns Header; 3) Task Area. Each of these will be explored in the sections that follow. There is also an Input/Message line between the Summary Area and Columns Header which needs no further explanation.

The main top screen is *generally* quite adaptive to changes in terminal dimensions under X-Windows. Other top screens may be less so, especially those with static text. It ultimately depends, however, on your particular window manager and terminal emulator. There may be occasions when their view of terminal size and current contents differs from top's view, which is always based on operating system calls.

Following any re-size operation, if a top screen is corrupted, appears incomplete or disordered, simply typing something innocuous like a punctuation character or cursor motion key will usually restore it. In extreme cases, the following sequence almost certainly will:

```
key/cmd objective
^Z    suspend top
fg    resume top
<Left> force a screenr edraw (if necessary)
```

But if the display is still corrupted, there is one more step you could try. Insert this command after top has been suspended but before resuming it.

```
key/cmd objective
reset restore yourterminal settings
```

Note: the width of top's display will be limited to 512 positions. Displaying all fields requires approximately 250 characters. Remaining screen width is usually allocated to any variable width columns currently visible. The variable width columns, such as COMMAND, are noted in topic 3a. DESCRIPTIONS of Fields. Actual output width may also be influenced by the -w switch, which is discussed in topic 1. COMMAND-LINE Options.

Lastly, some of top's screens or functions require the use of cursor motion keys like the standard arrow keys plus the Home, End, PgUp and PgDn keys. If your terminal or emulator does not provide those keys, the following combinations are accepted as alternatives:

key	equivalent-keys
Left	alt +h
Down	alt +j
Up	alt +k
Right	alt +l
Home	alt + ctrl +h
PgDn	alt + ctrl +j
PgUp	alt + ctrl +k
End	alt + ctrl +l

The **Up** and **Down** arrow keys have special significance when prompted for line input terminated with the <Enter> key. Those keys, or their aliases, can be used to retrieve previous input lines which can then be edited and re-input. And there are four additional keys available with line oriented input.

<i>key</i>	<i>special-significance</i>
Up	recall older strings for re-editing
Down	recall newer strings or erase entire line
Insert	toggle between insert and o vertype modes
Delete	character removed at cursor, moving others left
Home	jump to beginning of input line
End	jump to end of input line

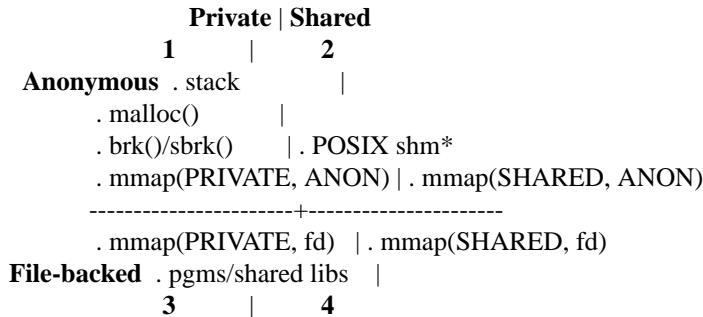
Linux Memory Types

For our purposes there are three types of memory, and one is optional. First is physical memory, a limited resource where code and data must reside when executed or referenced. Next is the optional swap file, where modified (dirty) memory can be saved and later retrieved if too many demands are made on physical memory. Lastly we have virtual memory, a nearly unlimited resource serving the following goals:

1. abstraction, free from physical memory addresses/limits
2. isolation, every process in a separate address space
3. sharing, a single mapping can serve multiple needs
4. flexibility, assign a virtual address to a file

Regardless of which of these forms memory may take, all are managed as pages (typically 4096 bytes) but expressed by default in top as KiB (kibibyte). The memory discussed under topic ‘2c. MEMORY Usage’ deals with physical memory and the swap file for the system as a whole. The memory reviewed in topic ‘3. FIELDS / Columns Display’ embraces all three memory types, but for individual processes.

For each such process, every memory page is restricted to a single quadrant from the table below. Both physical memory and virtual memory can include any of the four, while the swap file only includes #1 through #3. The memory in quadrant #4, when modified, acts as its own dedicated swap file.



The following may help in interpreting process level memory values displayed as scalable columns and discussed under topic ‘3a. DESCRIPTIONS of Fields’.

- %MEM – simply RES divided by total physical memory
- CODE – the ‘pgms’ portion of quadrant **3**
- DATA – the entire quadrant **1** portion of VIRT plus all explicit mmap file-backed pages of quadrant **3**
- RES – anything occupying physical memory which, beginning with Linux-4.5, is the sum of the following three fields:
 - RSan – quadrant **1** pages, which include any former quadrant **3** pages if modified
 - RSfd – quadrant **3** and quadrant **4** pages

RSsh – quadrant **2** pages
 RSlk – subset of RES which cannot be swapped out (any quadrant)
 SHR – subset of RES (excludes **1**, includes all **2 & 4**, some **3**)
 SWAP – potentially any quadrant except **4**
 USED – simply the sum of RES and SWAP
 VIRT – everything in-use and/or reserved (all quadrants)

Note: Even though program images and shared libraries are considered *private* to a process, they will be accounted for as *shared* (SHR) by the kernel.

1. COMMAND-LINE Options

The command-line syntax for top consists of:

-hv|bcEeHiOSs1 -d secs -n max -u|U user -p pids -o field -w [cols]

The typically mandatory switch ('-') and even whitespace are completely optional.

-h | -v :Help/Version

Show library version and the usage prompt, then quit.

-b :Batch-mode operation

Starts top in Batch mode, which could be useful for sending output from top to other programs or to a file. In this mode, top will not accept input and runs until the iterations limit you've set with the '-n' command-line option or until killed.

-c :Command-line/Program-name toggle

Starts top with the last remembered 'c' state reversed. Thus, if top was displaying command lines, now that field will show program names, and vice versa. See the 'c' interactive command for additional information.

-d :Delay-time interval as: **-d ss.t (secs.tenths)**

Specifies the delay between screen updates, and overrides the corresponding value in one's personal configuration file or the startup default. Later this can be changed with the 'd' or 's' interactive commands.

Fractional seconds are honored, but a negative number is not allowed. In all cases, however, such changes are prohibited if top is running in Secure mode, except for root (unless the 's' command-line option was used). For additional information on Secure mode see topic 6d. SYSTEM Restrictions File.

-e :Enforce-Task-Memory-Scaling as: **-e k|m|g|t|p**

Instructs top to force task area memory to be scaled as:

k – kibibytes

m – mebibytes

g – gibibytes

t – tebibytes

p – pebibytes

Later this can be changed with the 'e' command toggle.

-E :Enforce-Summary-Memory-Scaling as: -E k|m|g|t|p|e

Instructs top to force summary area memory to be scaled as:

- k – kibibytes
- m – mebibytes
- g – gibibytes
- t – tebibytes
- p – pebibytes
- e – exbibytes

Later this can be changed with the ‘E’ command toggle.

-H :Threads-mode operation

Instructs top to display individual threads. Without this command-line option a summation of all threads in each process is shown. Later this can be changed with the ‘H’ interactive command.

-i :Idle-process toggle

Starts top with the last remembered ‘i’ state reversed. When this toggle is *Off*, tasks that have not used any CPU since the last update will not be displayed. For additional information regarding this toggle see topic 4c. TASK AREA Commands, SIZE.

-n :Number-of-iterations limit as: -n number

Specifies the maximum number of iterations, or frames, top should produce before ending.

-o :Override-sort-field as: -o fieldname

Specifies the name of the field on which tasks will be sorted, independent of what is reflected in the configuration file. You can prepend a ‘+’ or ‘-’ to the field name to also override the sort direction. A leading ‘+’ will force sorting high to low, whereas a ‘-’ will ensure a low to high ordering.

This option exists primarily to support automated/scripted batch mode operation.

-O :Output-field-names

This option acts as a form of help for the above –o option. It will cause top to print each of the available field names on a separate line, then quit. Such names are subject to NLS (National Language Support) translation.

-p :Monitor-PIDs mode as: -pN1 -pN2 ... or -pN1,N2,N3 ...

Monitor only processes with specified process IDs. This option can be given up to 20 times, or you can provide a comma delimited list with up to 20 pids. Co-mingling both approaches is permitted.

A pid value of zero will be treated as the process id of the top program itself once it is running.

This is a command-line option only and should you wish to return to normal operation, it is not necessary to quit and restart top — just issue any of these interactive commands: ‘=’, ‘u’ or ‘U’.

The ‘p’, ‘u’ and ‘U’ command-line options are mutually exclusive.

-s :Secure-mode operation

Starts top with secure mode forced, even for root. This mode is far better controlled through a system configuration file (see topic 6. FILES).

-S :*Cumulative-time toggle*

Starts top with the last remembered ‘S’ state reversed. When Cumulative time mode is *On*, each process is listed with the cpu time that it and its dead children have used. See the ‘S’ interactive command for additional information regarding this mode.

-u | -U :*User-filter-mode* as: **-u | -U number or name**

Display only processes with a user id or user name matching that given. The ‘-u’ option matches on *effective* user whereas the ‘-U’ option matches on *any* user (real, effective, saved, or filesystem).

Prepending an exclamation point (!) to the user id or name instructs top to display only processes with users not matching the one provided.

The ‘p’, ‘u’ and ‘U’ command-line options are mutually exclusive.

-w :*Output-width-override* as: **-w [number]**

In Batch mode, when used without an argument top will format output using the COLUMNS= and LINES= environment variables, if set. Otherwise, width will be fixed at the maximum 512 columns. With an argument, output width can be decreased or increased (up to 512) but the number of rows is considered unlimited.

In normal display mode, when used without an argument top will *attempt* to format output using the COLUMNS= and LINES= environment variables, if set. With an argument, output width can only be decreased, not increased. Whether using environment variables or an argument with –w, when *not* in Batch mode actual terminal dimensions can never be exceeded.

Note: Without the use of this command-line option, output width is always based on the terminal at which top was invoked whether or not in Batch mode.

-1 :*Single/Separate-Cpu-States toggle*

Starts top with the last remembered Cpu States portion of the summary area reversed. Either all cpu information will be displayed in a single line or each cpu will be displayed separately, depending on the state of the NUMA Node command toggle (‘2’).

See the ‘1’ and ‘2’ interactive commands for additional information.

2. SUMMARY Display

Each of the following three areas are individually controlled through one or more interactive commands. See topic 4b. SUMMARY AREA Commands for additional information regarding these provisions.

2a. UPTIME and LOAD Averages

This portion consists of a single line containing:

- program** or **window** name, depending on display mode
- current time and length of time since last boot
- total number of users
- system load avg over the last 1, 5 and 15 minutes

2b. TASK and CPU States

This portion consists of a minimum of two lines. In an SMP environment, additional lines can reflect individual CPU state percentages.

Line 1 shows total **tasks** or **threads**, depending on the state of the Threads-mode toggle. That total is further classified as:

running; sleeping; stopped; zombie

Line 2 shows CPU state percentages based on the interval since the last refresh.

As a default, percentages for these individual categories are displayed. Where two labels are shown below, those for more recent kernel versions are shown first.

- us, user** : time running un-niced user processes
- sy, system** : time running kernel processes
- ni, nice** : time running niced user processes
- id, idle** : time spent in the kernel idle handler
- wa, IO-wait** : time waiting for I/O completion
- hi** : time spent servicing hardware interrupts
- si** : time spent servicing software interrupts
- st** : time stolen from this vm by the hypervisor

In the alternate cpu states display modes, beyond the first tasks/threads line, an abbreviated summary is shown consisting of these elements:

```
a b c d
%Cpu(s): 75.0/25.0 100[ ...]
```

Where: a) is the ‘user’ (us + ni) percentage; b) is the ‘system’ (sy + hi + si) percentage; c) is the total; and d) is one of two visual graphs of those representations. See topic 4b. SUMMARY AREA Commands and the ‘t’ command for additional information on that special 4-way toggle.

2c. MEMORY Usage

This portion consists of two lines which may express values in kibibytes (KiB) through exbibytes (EiB) depending on the scaling factor enforced with the ‘E’ interactive command.

As a default, Line 1 reflects physical memory, classified as:
total, free, used and buff/cache

Line 2 reflects mostly virtual memory, classified as:
total, free, used and avail (which is physical memory)

The **avail** number on line 2 is an estimation of physical memory available for starting new applications, without swapping. Unlike the **free** field, it attempts to account for readily reclaimable page cache and memory slabs. It is available on kernels 3.14, emulated on kernels 2.6.27+, otherwise the same as **free**.

In the alternate memory display modes, two abbreviated summary lines are shown consisting of these elements:

```
a b c
GiB Mem : 18.7/15.738 [ ...
GiB Swap: 0.0/7.999 [ ...]
```

Where: a) is the percentage used; b) is the total available; and c) is one of two visual graphs of those representations.

In the case of physical memory, the percentage represents the **total** minus the estimated **avail** noted above. The ‘Mem’ graph itself is divided between **used** and any remaining memory not otherwise accounted for by **avail**. See topic 4b. SUMMARY AREA Commands and the ‘m’ command for additional information on that special 4-way toggle.

This table may help in interpreting the scaled values displayed:

KiB = kibibyte = 1024 bytes
 MiB = mebibyte = 1024 KiB = 1,048,576 bytes
 GiB = gibibyte = 1024 MiB = 1,073,741,824 bytes
 TiB = tebibyte = 1024 GiB = 1,099,511,627,776 bytes
 PiB = pebibyte = 1024 TiB = 1,125,899,906,842,624 bytes
 EiB = exbibyte = 1024 PiB = 1,152,921,504,606,846,976 bytes

3. FIELDS / Columns

3a. DESCRIPTIONS of Fields

Listed below are top's available process fields (columns). They are shown in strict ascii alphabetical order. You may customize their position and whether or not they are displayable with the 'f' or 'F' (Fields Management) interactive commands.

Any field is selectable as the sort field, and you control whether they are sorted high-to-low or low-to-high. For additional information on sort provisions see topic 4c. TASK AREA Commands, SORTING.

The fields related to physical memory or virtual memory reference '(KiB)' which is the unsuffixed display mode. Such fields may, however, be scaled from KiB through PiB. That scaling is influenced via the 'e' interactive command or established for startup through a build option.

1. %CPU — CPU Usage

The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time.

In a true SMP environment, if a process is multi-threaded and top is *not* operating in Threads mode, amounts greater than 100% may be reported. You toggle Threads mode with the 'H' interactive command.

Also for multi-processor environments, if Irix mode is *Off*, top will operate in Solaris mode where a task's cpu usage will be divided by the total number of CPUs. You toggle Irix/Solaris modes with the 'T' interactive command.

Note: When running in forest view mode ('V') with children collapsed ('v'), this field will also include the CPU time of those unseen children. See topic 4c. TASK AREA Commands, CONTENT for more information regarding the 'V' and 'v' toggles.

2. %MEM — Memory Usage (RES)

A task's currently resident share of available physical memory.

See 'OVERVIEW, Linux Memory Types' for additional details.

3. CGNAME — Control Group Name

The name of the control group to which a process belongs, or '-' if not applicable for that process.

This will typically be the last entry in the full list of control groups as shown under the next heading (CGROUPS). And as is true there, this field is also variable width.

4. CGROUPS — Control Groups

The names of the control group(s) to which a process belongs, or '-' if not applicable for that process.

Control Groups provide for allocating resources (cpu, memory, network bandwidth, etc.) among installation-defined groups of processes. They enable fine-grained control over allocating, denying,

prioritizing, managing and monitoring those resources.

Many different hierarchies of cgroups can exist simultaneously on a system and each hierarchy is attached to one or more subsystems. A subsystem represents a single resource.

Note: The CGROUPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

5. CODE -- Code Size (KiB)

The amount of physical memory currently devoted to executable code, also known as the Text Resident Set size or TRS.

See ‘OVERVIEW, Linux Memory Types’ for additional details.

6. COMMAND -- Command Name or Command Line

Display the command line used to start a task or the name of the associated program. You toggle between command *line* and *name* with ‘c’, which is both a command-line option and an interactive command.

When you’ve chosen to display command lines, processes without a command line (like kernel threads) will be shown with only the program name in brackets, as in this example:

[kthreadd]

This field may also be impacted by the forest view display mode. See the ‘V’ interactive command for additional information regarding that mode.

Note: The COMMAND field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. This is especially true for this field when command lines are being displayed (the ‘c’ interactive command.) See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

7. DATA -- Data + Stack Size (KiB)

The amount of private memory *reserved* by a process. It is also known as the Data Resident Set or DRS. Such memory may not yet be mapped to physical memory (RES) but will always be included in the virtual memory (VIRT) amount.

See ‘OVERVIEW, Linux Memory Types’ for additional details.

8. ENVIRON -- Environment variables

Display all of the environment variables, if any, as seen by the respective processes. These variables will be displayed in their raw native order, not the sorted order you are accustomed to seeing with an unqualified ‘set’.

Note: The ENVIRON field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. This is especially true for this field. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

9. Flags -- Task Flags

This column represents the task's current scheduling flags which are expressed in hexadecimal notation and with zeros suppressed. These flags are officially documented in <linux/sched.h>.

10. GID -- Group Id

The *effective* group ID.

11. GROUP -- Group Name

The *effective* group name.

12. LXC -- Lxc Container Name

The name of the lxc container within which a task is running. If a process is not running inside a container, a dash ('-') will be shown.

13. NI -- Nice Value

The nice value of the task. A negative nice value means higher priority, whereas a positive nice value means lower priority. Zero in this field simply means priority will not be adjusted in determining a task's dispatch-ability.

14. NU -- Last known NUMA node

A number representing the NUMA node associated with the last used processor ('P'). When -1 is displayed it means that NUMA information is not available.

See the '2' and '3' interactive commands for additional NUMA provisions affecting the summary area.

15. OOMa -- Out of Memory Adjustment Factor

The value, ranging from -1000 to +1000, added to the current out of memory score (OOMs) which is then used to determine which task to kill when memory is exhausted.

16. OOMs -- Out of Memory Score

The value, ranging from 0 to +1000, used to select task(s) to kill when memory is exhausted. Zero translates to 'never kill' whereas 1000 means 'always kill'.

17. P -- Last used CPU (SMP)

A number representing the last used processor. In a true SMP environment this will likely change frequently since the kernel intentionally uses weak affinity. Also, the very act of running top may break this weak affinity and cause more processes to change CPUs more often (because of the extra demand for cpu time).

18. PGRP -- Process Group Id

Every process is member of a unique process group which is used for distribution of signals and by terminals to arbitrate requests for their input and output. When a process is created (forked), it becomes a member of the process group of its parent. By convention, this value equals the process ID (see PID) of the first member of a process group, called the process group leader.

19. PID -- Process Id

The task's unique process ID, which periodically wraps, though never restarting at zero. In kernel terms, it is a dispatchable entity defined by a task_struct.

This value may also be used as: a process group ID (see PGRP); a session ID for the session leader (see SID); a thread group ID for the thread group leader (see Tgid); and a TTY process group ID for the process group leader (see TPGID).

20. PPID -- Parent Process Id

The process ID (pid) of a task's parent.

21. PR -- Priority

The scheduling priority of the task. If you see 'rt' in this field, it means the task is running under real time scheduling priority.

Under linux, real time priority is somewhat misleading since traditionally the operating itself was not preemptible. And while the 2.6 kernel can be made mostly preemptible, it is not always so.

22. RES -- Resident Memory Size (KiB)

A subset of the virtual address space (VIRT) representing the non-swapped physical memory a task is currently using. It is also the sum of the RSan, RSfd and RSsh fields.

It can include private anonymous pages, private pages mapped to files (including program images and shared libraries) plus shared anonymous pages. All such memory is backed by the swap file represented separately under SWAP.

Lastly, this field may also include shared file-backed pages which, when modified, act as a dedicated swap file and thus will never impact SWAP.

See 'OVERVIEW, Linux Memory Types' for additional details.

23. RSan -- Resident Anonymous Memory Size (KiB)

A subset of resident memory (RES) representing private pages not mapped to a file.

24. RSfd -- Resident File-Backed Memory Size (KiB)

A subset of resident memory (RES) representing the implicitly shared pages supporting program images and shared libraries. It also includes explicit file mappings, both private and shared.

25. RSlk -- Resident Locked Memory Size (KiB)

A subset of resident memory (RES) which cannot be swapped out.

26. RSsh -- Resident Shared Memory Size (KiB)

A subset of resident memory (RES) representing the explicitly shared anonymous shm*/mmap pages.

27. RUID -- Real User Id

The *real* user ID.

28. RUSER -- Real User Name

The *real* user name.

29. S -- Process Status

The status of the task which can be one of:

D = uninterruptible sleep

I = idle

R = running
S = sleeping
T = stopped by job control signal
t = stopped by debugger during trace
Z = zombie

Tasks shown as running should be more properly thought of as ready to run -- their task_struct is simply represented on the Linux run-queue. Even without a true SMP machine, you may see numerous tasks in this state depending on top's delay interval and nice value.

30. **SHR** — Shared Memory Size (KiB)

A subset of resident memory (RES) that may be used by other processes. It will include shared anonymous pages and shared file-backed pages. It also includes private pages mapped to files representing program images and shared libraries.

See ‘OVERVIEW, Linux Memory Types’ for additional details.

31. **SID** — Session Id

A session is a collection of process groups (see PGRP), usually established by the login shell. A newly forked process joins the session of its creator. By convention, this value equals the process ID (see PID) of the first member of the session, called the session leader, which is usually the login shell.

32. **SUID** — Saved User Id

The *saved* user ID.

33. **SUPGIDS** — Supplementary Group IDs

The IDs of any supplementary group(s) established at login or inherited from a task’s parent. They are displayed in a comma delimited list.

Note: The SUPGIDS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

34. **SUPGRPS** — Supplementary Group Names

The names of any supplementary group(s) established at login or inherited from a task’s parent. They are displayed in a comma delimited list.

Note: The SUPGRPS field, unlike most columns, is not fixed-width. When displayed, it plus any other variable width columns will be allocated all remaining screen width (up to the maximum 512 characters). Even so, such variable width fields could still suffer truncation. See topic 5c. SCROLLING a Window for additional information on accessing any truncated data.

35. **SUSER** — Saved User Name

The *saved* user name.

36. **SWAP** — Swapped Size (KiB)

The formerly resident portion of a task’s address space written to the swap file when physical memory becomes over committed.

See ‘OVERVIEW, Linux Memory Types’ for additional details.

37. TGID — Thread Group Id

The ID of the thread group to which a task belongs. It is the PID of the thread group leader. In kernel terms, it represents those tasks that share an mm_struct.

38. TIME — CPU Time

Total CPU time the task has used since it started. When Cumulative mode is *On*, each process is listed with the cpu time that it and its dead children have used. You toggle Cumulative mode with ‘S’, which is both a command-line option and an interactive command. See the ‘S’ interactive command for additional information regarding this mode.

39. TIME+ — CPU Time, hundredths

The same as TIME, but reflecting more granularity through hundredths of a second.

40. TPGID — Tty Process Group Id

The process group ID of the foreground process for the connected tty, or -1 if a process is not connected to a terminal. By convention, this value equals the process ID (see PID) of the process group leader (see PGRP).

41. TTY — Controlling Tty

The name of the controlling terminal. This is usually the device (serial port, pty, etc.) from which the process was started, and which it uses for input or output. However, a task need not be associated with a terminal, in which case you’ll see ‘?’ displayed.

42. UID — User Id

The *effective* user ID of the task’s owner.

43. USED — Memory in Use (KiB)

This field represents the non-swapped physical memory a task is using (RES) plus the swapped out portion of its address space (SWAP).

See ‘OVERVIEW, Linux Memory Types’ for additional details.

44. USER — User Name

The *effective* user name of the task’s owner.

45. VIRT — Virtual Memory Size (KiB)

The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out and pages that have been mapped but not used.

See ‘OVERVIEW, Linux Memory Types’ for additional details.

46. WCHAN — Sleeping in Function

This field will show the name of the kernel function in which the task is currently sleeping. Running tasks will display a dash (‘-’) in this column.

47. nDRT — Dirty Pages Count

The number of pages that have been modified since they were last written to auxiliary storage. Dirty pages must be written to auxiliary storage before the corresponding physical memory location can be used for some other virtual page.

This field was deprecated with linux 2.6 and is always zero.

48. **nMaj** — Major Page Fault Count

The number of **major** page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A major page fault is when auxiliary storage access is involved in making that page available.

49. **nMin** — Minor Page Fault count

The number of **minor** page faults that have occurred for a task. A page fault occurs when a process attempts to read from or write to a virtual page that is not currently present in its address space. A minor page fault does not involve auxiliary storage access in making that page available.

50. **nTH** — Number of Threads

The number of threads associated with a process.

51. **nsIPC** — IPC namespace

The Inode of the namespace used to isolate interprocess communication (IPC) resources such as System V IPC objects and POSIX message queues.

52. **nsMNT** — MNT namespace

The Inode of the namespace used to isolate filesystem mount points thus offering different views of the filesystem hierarchy.

53. **nsNET** — NET namespace

The Inode of the namespace used to isolate resources such as network devices, IP addresses, IP routing, port numbers, etc.

54. **nsPID** — PID namespace

The Inode of the namespace used to isolate process ID numbers meaning they need not remain unique. Thus, each such namespace could have its own ‘init/systemd’ (PID #1) to manage various initialization tasks and reap orphaned child processes.

55. **nsUSER** — USER namespace

The Inode of the namespace used to isolate the user and group ID numbers. Thus, a process could have a normal unprivileged user ID outside a user namespace while having a user ID of 0, with full root privileges, inside that namespace.

56. **nsUTS** — UTS namespace

The Inode of the namespace used to isolate hostname and NIS domain name. UTS simply means “UNIX Time-sharing System”.

57. **vMj** — Major Page Fault Count Delta

The number of **major** page faults that have occurred since the last update (see nMaj).

58. **vMn** — Minor Page Fault Count Delta

The number of **minor** page faults that have occurred since the last update (see nMin).

3b. MANAGING Fields

After pressing the interactive command ‘f’ or ‘F’ (Fields Management) you will be presented with a screen showing: 1) the ‘current’ window name; 2) the designated sort field; 3) all fields in their current order along

with descriptions. Entries marked with an asterisk are the currently displayed fields, screen width permitting.

- As the on screen instructions indicate, you navigate among the fields with the **Up** and **Down** arrow keys. The PgUp, PgDn, Home and End keys can also be used to quickly reach the first or last available field.
- The **Right** arrow key selects a field for repositioning and the **Left** arrow key or the <**Enter**> key commits that field's placement.
- The 'd' key or the <**Space**> bar toggles a field's display status, and thus the presence or absence of the asterisk.
- The 's' key designates a field as the sort field. See topic 4c. TASK AREA Commands, SORTING for additional information regarding your selection of a sort field.
- The 'a' and 'w' keys can be used to cycle through all available windows and the 'q' or <**Esc**> keys exit Fields Management.

The Fields Management screen can also be used to change the 'current' window/field group in either full-screen mode or alternate-display mode. Whatever was targeted when 'q' or <**Esc**> was pressed will be made current as you return to the top display. See topic 5. ALTERNATE-DISPLAY Provisions and the 'g' interactive command for insight into 'current' windows and field groups.

Note: Any window that has been scrolled *horizontally* will be reset if any field changes are made via the Fields Management screen. Any *vertical* scrolled position, however, will not be affected. See topic 5c. SCROLLING a Window for additional information regarding vertical and horizontal scrolling.

4. INTERACTIVE Commands

Listed below is a brief index of commands within categories. Some commands appear more than once -- their meaning or scope may vary depending on the context in which they are issued.

4a. Global-Commands

<**Ent/Sp**> ?, =, 0,
A, B, d, E, e, g, h, H, I, k, q, r, s, W, X, Y, Z

4b. Summary-Area-Commands

C, l, t, m, 1, 2, 3, 4, !

4c. Task-Area-Commands

Appearance: b, J, j, x, y, z
Content: c, f, F, o, O, S, u, U, V, v
Size: #, i, n
Sorting: <, >, f, F, R

4d. Color-Mapping

<**Ret**>, a, B, b, H, M, q, S, T, w, z, 0 – 7

5b. Commands-for-Windows

–, _, =, +, A, a, g, G, w

5c. Scrolling-a-Window

C, Up, Dn, Left, Right, PgUp, PgDn, Home, End

5d. Searching-in-a-Window

L, &

4a. GLOBAL Commands

The global interactive commands are **always** available in both full–screen mode and alternate–display mode. However, some of these interactive commands are **not available** when running in Secure mode.

If you wish to know in advance whether or not your top has been secured, simply ask for help and view the system summary on the second line.

<Enter> or <Space> :Refresh-Display

These commands awaken top and following receipt of any input the entire display will be repainted. They also force an update of any hotplugged cpu or physical memory changes.

Use either of these keys if you have a large delay interval and wish to see current status,

? | h :Help

There are two help levels available. The first will provide a reminder of all the basic interactive commands. If top is secured, that screen will be abbreviated.

Typing ‘h’ or ‘?’ on that help screen will take you to help for those interactive commands applicable to alternate–display mode.

= :Exit-Display-Limits

Removes restrictions on what is shown. This command will reverse any ‘i’ (idle tasks), ‘n’ (max tasks) and ‘v’ (hide children) commands that might be active. It also provides for an exit from PID monitoring, User filtering, Other filtering, Locate processing and Combine Cpus mode.

Additionally, if the window has been scrolled it will be reset with this command.

0 :Zero-Suppress toggle

This command determines whether zeros are shown or suppressed for many of the fields in a task window. Fields like UID, GID, NI, PR or P are not affected by this toggle.

A :Alternate-Display-Mode toggle

This command will switch between full–screen mode and alternate–display mode. See topic 5. ALTERNATE–DISPLAY Provisions and the ‘g’ interactive command for insight into ‘current’ windows and field groups.

B :Bold-Disable/Enable toggle

This command will influence use of the bold terminfo capability and alters **both** the summary area and task area for the ‘current’ window. While it is intended primarily for use with dumb terminals, it can be applied anytime.

Note: When this toggle is *On* and top is operating in monochrome mode, the **entire display** will appear as normal text. Thus, unless the ‘x’ and/or ‘y’ toggles are using reverse for emphasis, there will be no visual confirmation that they are even on.

*** d | s :Change-Delay-Time-interval**

You will be prompted to enter the delay time, in seconds, between display updates.

Fractional seconds are honored, but a negative number is not allowed. Entering 0 causes (nearly) continuous updates, with an unsatisfactory display as the system and tty driver try to keep up with top’s demands. The delay value is inversely proportional to system loading, so set it with care.

If at any time you wish to know the current delay time, simply ask for help and view the system summary on the second line.

E :Enforce-Summary-Memory-Scale in Summary Area

With this command you can cycle through the available summary area memory scaling which ranges from KiB (kilobytes or 1,024 bytes) through EiB (exabytes or 1,152,921,504,606,846,976 bytes).

If you see a ‘+’ between a displayed number and the following label, it means that top was forced to truncate some portion of that number. By raising the scaling factor, such truncation can be avoided.

e :Enforce-Task-Memory-Scale in Task Area

With this command you can cycle through the available task area memory scaling which ranges from KiB (kilobytes or 1,024 bytes) through PiB (pebibytes or 1,125,899,906,842,624 bytes).

While top will try to honor the selected target range, additional scaling might still be necessary in order to accommodate current values. If you wish to see a more homogeneous result in the memory columns, raising the scaling range will usually accomplish that goal. Raising it too high, however, is likely to produce an all zero result which cannot be suppressed with the ‘0’ interactive command.

g :Choose-Another-Window/Field-Group

You will be prompted to enter a number between 1 and 4 designating the field group which should be made the ‘current’ window. You will soon grow comfortable with these 4 windows, especially after experimenting with alternate-display mode.

H :Threads-mode toggle

When this toggle is *On*, individual threads will be displayed for all processes in all visible task windows. Otherwise, top displays a summation of all threads in each process.

I :Irix/Solaris-Mode toggle

When operating in Solaris mode (‘I’ toggled *Off*), a task’s cpu usage will be divided by the total number of CPUs. After issuing this command, you’ll be told the new state of this toggle.

* **k** :Kill-a-task

You will be prompted for a PID and then the signal to send.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displayed). A PID value of zero means the top program itself.

The default signal, as reflected in the prompt, is SIGTERM. However, you can send any signal, via number or name.

If you wish to abort the kill process, do one of the following depending on your progress:

- 1) at the pid prompt, type an invalid number
- 2) at the signal prompt, type 0 (or any invalid signal)
- 3) at any prompt, type <Esc>

q :Quit

*** r :Renice-a-Task**

You will be prompted for a PID and then the value to nice it to.

Entering no PID or a negative number will be interpreted as the default shown in the prompt (the first task displayed). A PID value of zero means the top program itself.

A positive nice value will cause a process to lose priority. Conversely, a negative nice value will cause a process to be viewed more favorably by the kernel. As a general rule, ordinary users can only increase the nice value and are prevented from lowering it.

If you wish to abort the renice process, do one of the following depending on your progress:

- 1) at the pid prompt, type an invalid number
- 2) at the nice prompt, type <Enter> with no input
- 3) at any prompt, type <Esc>

W :Write-the-Configuration-File

This will save all of your options and toggles plus the current display mode and delay time. By issuing this command just before quitting top, you will be able restart later in exactly that same state.

X :Extra-Fixed-Width

Some fields are fixed width and not scalable. As such, they are subject to truncation which would be indicated by a '+' in the last position.

This interactive command can be used to alter the widths of the following fields:

<i>field</i>	<i>default</i>	<i>field</i>	<i>default</i>	<i>field</i>	<i>default</i>
GID	5	GROUP	8	WCHAN	10
RUID	5	LXC	8	nsIPC	10
SUID	5	RUSER	8	nsMNT	10
UID	5	SUSER	8	nsNET	10
		TTY	8	nsPID	10
		USER	8	nsUSER	10
				nsUTS	10

You will be prompted for the amount to be added to the default widths shown above. Entering zero forces a return to those defaults.

If you enter a negative number, top will automatically increase the column size as needed until there is no more truncated data. You can accelerate this process by reducing the delay interval or holding down the <Space> bar.

Note: Whether explicitly or automatically increased, the widths for these fields are never decreased by top. To narrow them you must specify a smaller number or restore the defaults.

Y :Inspect-Other-Output

After issuing the 'Y' interactive command, you will be prompted for a target PID. Typing a value or accepting the default results in a separate screen. That screen can be used to view a variety of files or piped command output while the normal top iterative display is paused.

Note: This interactive command is only fully realized when supporting entries have been manually added to the end of the top configuration file. For details on creating those entries, see topic 6b. ADDING INSPECT Entries.

Most of the keys used to navigate the Inspect feature are reflected in its header prologue. There are, however, additional keys available once you have selected a particular file or command. They are familiar to anyone who has used the pager ‘less’ and are summarized here for future reference.

<i>key</i>	<i>function</i>
=	alternate status-line, file or pipeline
/	find, equivalent to ‘L’ locate
n	find next, equivalent to ‘&’ locate next
<Space>	scroll down, equivalent to <PgDn>
b	scroll up, equivalent to <PgUp>
g	first line, equivalent to <Home>
G	last line, equivalent to <End>

Z :Change-Color-Mapping

This key will take you to a separate screen where you can change the colors for the ‘current’ window, or for all windows. For details regarding this interactive command see topic 4d. COLOR Mapping.

- * The commands shown with an asterisk (“*”) are not available in Secure mode, nor will they be shown on the level-1 help screen.

4b. SUMMARY AREA Commands

The summary area interactive commands are **always available** in both full-screen mode and alternate-display mode. They affect the beginning lines of your display and will determine the position of messages and prompts.

These commands always impact just the ‘current’ window/field group. See topic 5. ALTERNATE-DISPLAY Provisions and the ‘g’ interactive command for insight into ‘current’ windows and field groups.

C :Show-scroll-coordinates toggle

Toggle an informational message which is displayed whenever the message line is not otherwise being used. For additional information see topic 5c. SCROLLING a Window.

I :Load-Average/Uptime toggle

This is also the line containing the program name (possibly an alias) when operating in full-screen mode or the ‘current’ window name when operating in alternate-display mode.

t :Task/Cpu-States toggle

This command affects from 2 to many summary area lines, depending on the state of the ‘1’, ‘2’ or ‘3’ command toggles and whether or not top is running under true SMP.

This portion of the summary area is also influenced by the ‘H’ interactive command toggle, as reflected in the total label which shows either Tasks or Threads.

This command serves as a 4-way toggle, cycling through these modes:

1. detailed percentages by category
2. abbreviated user/system and total % + bar graph
3. abbreviated user/system and total % + block graph
4. turn off task and cpu states display

When operating in either of the graphic modes, the display becomes much more meaningful when

individual CPUs or NUMA nodes are also displayed. See the the ‘1’, ‘2’ and ‘3’ commands below for additional information.

m :Memory/Swap-Usage toggle

This command affects the two summary area lines dealing with physical and virtual memory.

This command serves as a 4-way toggle, cycling through these modes:

1. detailed percentages by memory type
2. abbreviated % used/total available + bar graph
3. abbreviated % used/total available + block graph
4. turn off memory display

1 :Single/Separate-Cpu-States toggle

This command affects how the ‘t’ command’s Cpu States portion is shown. Although this toggle exists primarily to serve massively-parallel SMP machines, it is not restricted to solely SMP environments.

When you see ‘%Cpu(s):’ in the summary area, the ‘1’ toggle is *On* and all cpu information is gathered in a single line. Otherwise, each cpu is displayed separately as: ‘%Cpu0, %Cpu1, ...’ up to available screen height.

2 :NUMA-Nodes/Cpu-Summary toggle

This command toggles between the ‘1’ command cpu summary display (only) or a summary display plus the cpu usage statistics for each NUMA Node. It is only available if a system has the requisite NUMA support.

3 :Expand-NUMA-Node

You will be invited to enter a number representing a NUMA Node. Thereafter, a node summary plus the statistics for each cpu in that node will be shown until the ‘1’, ‘2’ or ‘4’ command toggle is pressed. This interactive command is only available if a system has the requisite NUMA support.

4 :Display-Cpus-Two-Abreast

This command turns the ‘1’ toggle *Off* for individual cpu display but prints the results two abreast. It requires a terminal with a minimum width of 80 columns. If a terminal’s width is decreased below the minimum while top is running, top reverts to the normal ‘1’ toggle *Off* state.

To avoid truncation when displaying detailed cpu statiscts, as opposed to the graphic representations, a minimum width of 165 columns would be required.

! :Combine-Cpus-Mode

This command toggle is intended for massively parallel SMP environments where, even with the ‘4’ command toggle, not all processors can be displayed. With each press of ‘!’ the number of additional cpu’s combined is doubled thus reducing the total number of cpu lines displayed.

For example, with the first press of ‘!’ one additional cpu will be combined and displayed as ‘0-1, 2-3, ...’ instead of the normal ‘%Cpu0, %Cpu1, %Cpu2, %Cpu3, ...’. With a second ‘!’ command toggle two additional cpus are combined and shown as ‘0-2, 3-5, ...’. Then the third ‘!’ press, combining four additional cpus, shows as ‘0-4, 5-9, ...’, etc.

Such progression continues until individual cpus are again displayed and impacts both the ‘1’ and ‘4’ toggles (one or two columns). Use the ‘=’ command to exit **Combine Cpus** mode.

Note: If the entire summary area has been toggled *Off* for any window, you would be left with just the **message line**. In that way, you will have maximized available task rows but (temporarily) sacrificed the program name in full-screen mode or the ‘current’ window name when in alternate-display mode.

4c. TASK AREA Commands

The task area interactive commands are **always** available in full-screen mode.

The task area interactive commands are **never available** in alternate-display mode *if* the ‘current’ window’s task display has been toggled *Off* (see topic 5. ALTERNATE-DISPLAY Provisions).

APPEARANCE of task window

J :*Justify-Numeric-Columns* toggle

Alternates between right-justified (the default) and left-justified numeric data. If the numeric data completely fills the available column, this command toggle may impact the column header only.

j :*Justify-Character-Columns* toggle

Alternates between left-justified (the default) and right-justified character data. If the character data completely fills the available column, this command toggle may impact the column header only.

The following commands will also be influenced by the state of the global ‘B’ (bold enable) toggle.

b :*Bold/Reverse* toggle

This command will impact how the ‘x’ and ‘y’ toggles are displayed. It may also impact the summary area when a bar graph has been selected for cpu states or memory usage via the ‘t’ or ‘m’ toggles.

x :*Column-Highlight* toggle

Changes highlighting for the current sort field. If you forget which field is being sorted this command can serve as a quick visual reminder, providing the sort field is being displayed. The sort field might *not* be visible because:

- 1) there is insufficient *Screen Width*
- 2) the ‘f’ interactive command turned it *Off*

Note: Whenever Searching and/or Other Filtering is active in a window, column highlighting is temporarily disabled. See the notes at the end of topics 5d. SEARCHING and 5e. FILTERING for an explanation why.

y :*Row-Highlight* toggle

Changes highlighting for “running” tasks. For additional insight into this task state, see topic 3a. DESCRIPTIONS of Fields, the ‘S’ field (Process Status).

Use of this provision provides important insight into your system’s health. The only costs will be a few additional tty escape sequences.

z :*Color/Monochrome* toggle

Switches the ‘current’ window between your last used color scheme and the older form of black-on-white or white-on-black. This command will alter **both** the summary area and task area but does not affect the state of the ‘x’, ‘y’ or ‘b’ toggles.

CONTENT of task window**c** :*Command-Line/Program-Name* toggle

This command will be honored whether or not the COMMAND column is currently visible. Later, should that field come into view, the change you applied will be seen.

f | F :*Fields-Management*

These keys display a separate screen where you can change which fields are displayed, their order and also designate the sort field. For additional information on these interactive commands see topic 3b. MANAGING Fields.

o | O :*Other-Filtering*

You will be prompted for the selection criteria which then determines which tasks will be shown in the ‘current’ window. Your criteria can be made case sensitive or case can be ignored. And you determine if top should include or exclude matching tasks.

See topic 5e. FILTERING in a window for details on these and additional related interactive commands.

S :*Cumulative-Time-Mode* toggle

When Cumulative mode is *On*, each process is listed with the cpu time that it and its dead children have used.

When *Off*, programs that fork into many separate tasks will appear less demanding. For programs like ‘init’ or a shell this is appropriate but for others, like compilers, perhaps not. Experiment with two task windows sharing the same sort field but with different ‘S’ states and see which representation you prefer.

After issuing this command, you’ll be informed of the new state of this toggle. If you wish to know in advance whether or not Cumulative mode is in effect, simply ask for help and view the window summary on the second line.

u | U :*Show-Specific-User-Only*

You will be prompted for the **uid** or **name** of the user to display. The –u option matches on **effective** user whereas the –U option matches on **any** user (real, effective, saved, or filesystem).

Thereafter, in that task window only matching users will be shown, or possibly no processes will be shown. Prepending an exclamation point (!) to the user id or name instructs top to display only processes with users not matching the one provided.

Different task windows can be used to filter different users. Later, if you wish to monitor all users again in the ‘current’ window, re-issue this command but just press <Enter> at the prompt.

V :*Forest-View-Mode* toggle

In this mode, processes are reordered according to their parents and the layout of the COMMAND column resembles that of a tree. In forest view mode it is still possible to toggle between program name and command line (see the ‘c’ interactive command) or between processes and threads (see the ‘H’ interactive command).

Note: Typing any key affecting the sort order will exit forest view mode in the ‘current’ window. See topic 4c. TASK AREA Commands, SORTING for information on those keys.

v :Hide/Show-Children toggle

When in forest view mode, this key serves as a toggle to collapse or expand the children of a parent.

The toggle is applied against the first (topmost) process in the ‘current’ window. See topic 5c. SCROLLING a Window for additional information regarding vertical scrolling.

If the target process has not forked any children, this key has no effect. It also has no effect when not in forest view mode.

SIZE of task window**i :Idle-Process toggle**

Displays all tasks or just active tasks. When this toggle is *Off*, tasks that have not used any CPU since the last update will not be displayed. However, due to the granularity of the %CPU and TIME+ fields, some processes may still be displayed that *appear* to have used *no* CPU.

If this command is applied to the last task display when in alternate–display mode, then it will not affect the window’s size, as all prior task displays will have already been painted.

n | # :Set-Maximum-Tasks

You will be prompted to enter the number of tasks to display. The lesser of your number and available screen rows will be used.

When used in alternate–display mode, this is the command that gives you precise control over the size of each currently visible task display, except for the very last. It will not affect the last window’s size, as all prior task displays will have already been painted.

Note: If you wish to increase the size of the last visible task display when in alternate–display mode, simply decrease the size of the task display(s) above it.

SORTING of task window

For compatibility, this top supports most of the former top sort keys. Since this is primarily a service to former top users, these commands do not appear on any help screen.

command	sorted-field	supported
A	start time (non-display)	No
M	%MEM	Yes
N	PID	Yes
P	%CPU	Yes
T	TIME+	Yes

Before using any of the following sort provisions, top suggests that you temporarily turn on column highlighting using the ‘x’ interactive command. That will help ensure that the actual sort environment matches your intent.

The following interactive commands will **only** be honored when the current sort field is **visible**. The sort field might *not* be visible because:

- 1) there is insufficient *Screen Width*
- 2) the ‘f’ interactive command turned it *Off*

< :Move-Sort-Field-Left

Moves the sort column to the left unless the current sort field is the first field being displayed.

> :Move-Sort-Field-Right

Moves the sort column to the right unless the current sort field is the last field being displayed.

The following interactive commands will **always** be honored whether or not the current sort field is visible.

f | F :Fields-Management

These keys display a separate screen where you can change which field is used as the sort column, among other functions. This can be a convenient way to simply verify the current sort field, when running top with column highlighting turned *Off*.

R :Reverse/Normal-Sort-Field toggle

Using this interactive command you can alternate between high-to-low and low-to-high sorts.

Note: Field sorting uses internal values, not those in column display. Thus, the TTY and WCHAN fields will violate strict ASCII collating sequence.

4d. COLOR Mapping

When you issue the ‘Z’ interactive command, you will be presented with a separate screen. That screen can be used to change the colors in just the ‘current’ window or in all four windows before returning to the top display.

The following interactive commands are available.

4 upper case letters to select a **target**

8 numbers to select a **color**

normal toggles available

B :bold disable/enable

b :running tasks "bold"/reverse

z :color/mono

other commands available

a/w :apply, then go to next/prior

<Enter> :apply and exit

q :abandon current changes and exit

If you use ‘a’ or ‘w’ to cycle the targeted window, you will have applied the color scheme that was displayed when you left that window. You can, of course, easily return to any window and reapply different colors or turn colors *Off* completely with the ‘z’ toggle.

The Color Mapping screen can also be used to change the ‘current’ window/field group in either full-screen mode or alternate-display mode. Whatever was targeted when ‘q’ or <Enter> was pressed will be made current as you return to the top display.

5. ALTERNATE-DISPLAY Provisions

5a. WINDOWS Overview

Field Groups/Windows:

In full-screen mode there is a single window represented by the entire screen. That single window can still be changed to display 1 of 4 different **field groups** (see the ‘g’ interactive command, repeated below). Each of the 4 field groups has a unique separately configurablesummary ar ea and its own

configurable **task area**.

In alternate–display mode, those 4 underlying field groups can now be made visible simultaneously, or can be turned *Off* individually at your command.

The summary area will always exist, even if it's only the message line. At any given time only *one* summary area can be displayed. However, depending on your commands, there could be from *zero* to *four* separate task displays currently showing on the screen.

Current Window:

The ‘current’ window is the window associated with the summary area and the window to which task related commands are always directed. Since in alternate–display mode you can toggle the task display *Off*, some commands might be restricted for the ‘current’ window.

A further complication arises when you have toggled the first summary area line *Off*. With the loss of the window name (the ‘l’ toggled line), you’ll not easily know what window is the ‘current’ window.

5b. COMMANDS for Windows

- | _ :Show/Hide-Window(s) toggles

The ‘-’ key turns the ‘current’ window’s task display *On* and *Off*. When *On*, that task area will show a minimum of the columns header you’ve established with the ‘f’ interactive command. It will also reflect any other task area options/toggles you’ve applied yielding zero or more tasks.

The ‘_’ key does the same for all task displays. In other words, it switches between the currently visible task display(s) and any task display(s) you had toggled *Off*. If all 4 task displays are currently visible, this interactive command will leave the summary area as the only display element.

* = | + :Equalize/Reset-Window(s)

The ‘=’ key forces the ‘current’ window’s task display to be visible. It also reverses any active ‘i’ (idle tasks), ‘n’ (max tasks), ‘u/U’ (user filter), ‘o/O’ (other filter), ‘v’ (hide children), ‘L’ (locate) and ‘!’ (combine cpus) commands. Also, if the window had been scrolled, it will be reset with this command. See topic 5c. SCROLLING a Window for additional information regarding vertical and horizontal scrolling.

The ‘+’ key does the same for all windows. The four task displays will reappear, evenly balanced, while retaining any customizations previously applied beyond those noted for the ‘=’ command toggle.

* **A** :Alternate-Display-Mode toggle

This command will switch between full–screen mode and alternate–display mode.

The first time you issue this command, all four task displays will be shown. Thereafter when you switch modes, you will see only the task display(s) you’ve chosen to make visible.

* **a** | **w** :Next-Window-Forward/Backward

This will change the ‘current’ window, which in turn changes the window to which commands are directed. These keys act in a circular fashion so you can reach any desired window using either key.

Assuming the window name is visible (you have not toggled ‘l’ *Off*), whenever the ‘current’ window name loses its emphasis/color, that’s a reminder the task display is *Off* and many

commands will be restricted.

*** g :Choose-Another-Window/Field-Group**

You will be prompted to enter a number between 1 and 4 designating the field group which should be made the ‘current’ window.

In full–screen mode, this command is necessary to alter the ‘current’ window. In alternate–display mode, it is simply a less convenient alternative to the ‘a’ and ‘w’ commands.

G :Change-Window/Field-Group-Name

You will be prompted for a new name to be applied to the ‘current’ window. It does not require that the window name be visible (the ‘l’ toggle to be *On*).

- * The interactive commands shown with an asterisk (“*”) have use beyond alternate–display mode.
- =, A, g are always available
- a, w act the same with color mapping
and fields management

5c. SCROLLING a Window

Typically a task window is a partial view into a system’s total tasks/threads which shows only some of the available fields/columns. With these scrolling keys, you can move that view vertically or horizontally to reveal any desired task or column.

Up,PgUp :Scroll-Tasks

Move the view up toward the first task row, until the first task is displayed at the top of the ‘current’ window. The *Up* arrow key moves a single line while *PgUp* scrolls the entire window.

Down,PgDn :Scroll-Tasks

Move the view down toward the last task row, until the last task is the only task displayed at the top of the ‘current’ window. The *Down* arrow key moves a single line while *PgDn* scrolls the entire window.

Left,Right :Scroll-Columns

Move the view of displayable fields horizontally one column at a time.

Note: As a reminder, some fields/columns are not fixed-width but allocated all remaining screen width when visible. When scrolling right or left, that feature may produce some unexpected results initially.

Additionally, there are special provisions for any variable width field when positioned as the last displayed field. Once that field is reached via the right arrow key, and is thus the only column shown, you can continue scrolling horizontally within such a field. See the ‘C’ interactive command below for additional information.

Home :Jump-to-Home-Position

Reposition the display to the un-scrolled coordinates.

End :Jump-to-End-Position

Reposition the display so that the rightmost column reflects the last displayable field and the bottom task row represents the last task.

Note: From this position it is still possible to scroll *down* and *right* using the arrow keys. This is true until a single column and a single task is left as the only display element.

C :Show-scroll-coordinates toggle

Toggle an informational message which is displayed whenever the message line is not otherwise being used. That message will take one of two forms depending on whether or not a variable width column has also been scrolled.

scroll coordinates: y = n/n (tasks), x = n/n (fields)

scroll coordinates: y = n/n (tasks), x = n/n (fields) + nn

The coordinates shown as **n/n** are relative to the upper left corner of the ‘current’ window. The additional ‘+ nn’ represents the displacement into a variable width column when it has been scrolled horizontally. Such displacement occurs in normal 8 character tab stop amounts via the right and left arrow keys.

y = n/n (tasks)

The first **n** represents the topmost visible task and is controlled by scrolling keys. The second **n** is updated automatically to reflect total tasks.

x = n/n (fields)

The first **n** represents the leftmost displayed column and is controlled by scrolling keys. The second **n** is the total number of displayable fields and is established with the ‘f’ interactive command.

The above interactive commands are **always** available in full-screen mode but **never** available in alternate-display mode if the ‘current’ window’s task display has been toggled *Off*.

Note: When any form of filtering is active, you can expect some slight aberrations when scrolling since not all tasks will be visible. This is particularly apparent when using the Up/Down arrow keys.

5d. SEARCHING in a Window

You can use these interactive commands to locate a task row containing a particular value.

L :Locate-a-string

You will be prompted for the case-sensitive string to locate starting from the current window coordinates. There are no restrictions on search string content.

Searches are not limited to values from a single field or column. All of the values displayed in a task row are allowed in a search string. You may include spaces, numbers, symbols and even forest view artwork.

Keying <Enter> with no input will effectively disable the ‘&’ key until a new search string is entered.

& :Locate-next

Assuming a search string has been established, top will attempt to locate the next occurrence.

When a match is found, the current window is repositioned vertically so the task row containing that string is first. The scroll coordinates message can provide confirmation of such vertical repositioning (see the ‘C’ interactive command). Horizontal scrolling, however, is never altered via searching.

The availability of a matching string will be influenced by the following factors.

- a. Which fields are displayable from the total available,
see topic 3b. MANAGING Fields.
- b. Scrolling a window vertically and/or horizontally,
see topic 5c. SCROLLING a Window.
- c. The state of the command/command-line toggle,
see the ‘c’ interactive command.
- d. The stability of the chosen sort column,
for example PID is good but %CPU bad.

If a search fails, restoring the ‘current’ window home (unscrolled) position, scrolling horizontally, displaying command-lines or choosing a more stable sort field could yet produce a successful ‘&’ search.

The above interactive commands are **always** available in full-screen mode but **never** available in alternate-display mode if the ‘current’ window’s task display has been toggled *Off*.

Note: Whenever a Search is active in a window, top will turn column highlighting *Off* to prevent false matches on internal non-display escape sequences. Such highlighting will be restored when a window’s search string is empty. See the ‘x’ interactive command for additional information on sort column highlighting.

5e. FILTERING in a Window

You can use this ‘Other Filter’ feature to establish selection criteria which will then determine which tasks are shown in the ‘current’ window. Such filters can be made persistent if preserved in the rfile via the ‘W’ interactive command.

Establishing a filter requires: 1) a field name; 2) an operator; and 3) a selection value, as a minimum. This is the most complex of top’s user input requirements so, when you make a mistake, command recall will be your friend. Remember the Up/Down arrow keys or their aliases when prompted for input.

Filter Basics

1. field names are case sensitive and spelled as in the header
2. selection values need not comprise the full displayed field
3. a selection is either case insensitive or sensitive to case
4. the default is inclusion, prepending ‘!’ denotes exclusions
5. multiple selection criteria can be applied to a task window
6. inclusion and exclusion criteria can be used simultaneously
7. the 1 equality and 2 relational filters can be freely mixed
8. separate unique filters are maintained for each task window

If a field is not turned on or is not currently in view, then your selection criteria will not affect the display. Later, should a filtered field become visible, the selection criteria will then be applied.

Keyboard Summary

o :*Other-Filter* (lower case)

You will be prompted to establish a filter that **ignores case** when matching.

O :*Other-Filter* (upper case)

You will be prompted to establish a **case sensitive** filter.

^O :*Show-Active-Filters* (Ctrl key + ‘o’)

This can serve as a reminder of which filters are active in the ‘current’ window. A summary will be shown on the message line until you press the <Enter> key.

= :*Reset-Filtering* in current window

This clears all of your selection criteria in the ‘current’ window. It also has additional impact so please see topic 4a. GLOBAL Commands.

+ :*Reset-Filtering* in all windows

This clears the selection criteria in all windows, assuming you are in alternate–display mode. As with the ‘=’ interactive command, it too has additional consequences so you might wish to see topic 5b. COMMANDS for Windows.

Input Requirements

When prompted for selection criteria, the data you provide must take one of two forms. There are 3 required pieces of information, with a 4th as optional. These examples use spaces for clarity but your input generally would not.

```
#1      #2 #3      ( required )
Field-Name ? include-if-value
! Field-Name ? exclude-if-value
#4                  ( optional )
```

Items #1, #3 and #4 should be self-explanatory. Item#2 represents both a required *delimiter* and the *operator* which must be one of either equality (‘=’) or relation (‘<’ or ‘>’).

The ‘=’ equality operator requires only a partial match and that can reduce your ‘if–value’ input requirements. The ‘>’ or ‘<’ relational operators always employ string comparisons, even with numeric fields. They are designed to work with a field’s default *justification* and with homogeneous data. When some field’s numeric amounts have been subjected to *scaling* while others have not, that data is no longer homogeneous.

If you establish a relational filter and you **have** changed the default Numeric or Character *justification*, that filter is likely to fail. When a relational filter is applied to a memory field and you **have not** changed the *scaling*, it may produce misleading results. This happens, for example, because ‘100.0m’ (MiB) would appear greater than ‘1.000g’ (GiB) when compared as strings.

If your filtered results appear suspect, simply altering justification or scaling may yet achieve the desired objective. See the ‘j’, ‘J’ and ‘e’ interactive commands for additional information.

Potential Problems

These **GROUP** filters could produce the exact same results or the second one might not display anything at all, just a blank task window.

```
GROUP=root      ( only the same results when )
GROUP=ROOT     ( invoked via lower case 'o' )
```

Either of these **RES** filters might yield inconsistent and/or misleading results, depending on the current memory scaling factor. Or both filters could produce the exact same results.

```
RES>9999      ( only the same results when )
!RES<10000    ( memory scaling is at 'KiB' )
```

This **nMin** filter illustrates a problem unique to scalable fields. This particular field can display a maximum of 4 digits, beyond which values are automatically scaled to KiB or above. So while amounts greater than 9999 exist, they will appear as 2.6m, 197k, etc.

`nMin>9999` (always a blank task window)

Potential Solutions

These examples illustrate how Other Filtering can be creatively applied to achieve almost any desired result. Single quotes are sometimes shown to delimit the spaces which are part of a filter or to represent a request for status (^O) accurately. But if you used them with if-values in real life, no matches would be found.

Assuming field **nTH** is displayed, the first filter will result in only multi-threaded processes being shown. It also reminds us that a trailing space is part of every displayed field. The second filter achieves the exact same results with less typing.

<code>!nTH=' 1'</code>	(' for clarity only)
<code>nTH>1</code>	(same with less i/p)

With Forest View mode active and the **COMMAND** column in view, this filter effectively collapses child processes so that just 3 levels are shown.

`!COMMAND=' -'` (' for clarity only)

The final two filters appear as in response to the status request key (^O). In reality, each filter would have required separate input. The **PR** example shows the two concurrent filters necessary to display tasks with priorities of 20 or more, since some might be negative. Then by exploiting trailing spaces, the **nMin** series of filters could achieve the failed '9999' objective discussed above.

<code>'PR>20' + '!PR=-'</code>	(2 for right result)
<code>'!nMin=0' + '!nMin=1' + '!nMin=2' + '!nMin=3' ...</code>	

Note: Whenever Other Filtering is active in a window, top will turn column highlighting *Off* to prevent false matches on internal non-display escape sequences. Such highlighting will be restored when a window is no longer subject to filtering. See the 'x' interactive command for additional information on sort column highlighting.

6. FILES

6a. PERSONAL Configuration File

This file is created or updated via the 'W' interactive command.

The legacy version is written as '\$HOME/.your-name-4-top' + 'rc' with a leading period.

A newly created configuration file is written as procps/your-name-4-top' + 'rc' without a leading period. The procps directory will be subordinate to either \$XDG_CONFIG_HOME when set as an absolute path or the \$HOME/.config directory.

While not intended to be edited manually, here is the general layout:

```
global # line 1: the program name/alias notation
      # line 2: id,altscr,irixps,delay,curwin
per ea # line a: winname,fieldscur
window # line b: winflags,sortindx,maxtasks,etc
      # line c: summclr,msgsclr,headclr,taskclr
global # line 15: additional miscellaneous settings
      # any remaining lines are devoted to optional
      # active 'other filters' discussed in section 5e above
      # plus 'inspect' entries discussed in section 6b below
```

If a valid absolute path to the rcf file cannot be established, customizations made to a running top will be impossible to preserve.

6b. ADDING INSPECT Entries

To exploit the ‘Y’ interactive command, you must add entries at the **end** of the top personal configuration file. Such entries simply reflect a file to be read or command/pipeline to be executed whose results will then be displayed in a separate scrollable, searchable window.

If you don’t know the location or name of your top rcf file, use the ‘W’ interactive command to rewrite it and note those details.

Inspect entries can be added with a redirected echo or by editing the configuration file. Redirecting an echo risks overwriting the rcf file should it replace (>) rather than append (>>) to that file. Conversely, when using an editor care must be taken not to corrupt existing lines, some of which will contain unprintable data or unusual characters.

Those Inspect entries beginning with a ‘#’ character are ignored, regardless of content. Otherwise they consist of the following 3 elements, each of which *must* be separated by a tab character (thus 2 ‘\t’ total):

- .type: literal ‘file’ or ‘pipe’
- .name: selection shown on the Inspect screen
- .fmts: string representing a path or command

The two types of Inspect entries are *not* interchangeable. Those designated ‘file’ will be accessed using fopen and must reference a single file in the ‘.fmts’ element. Entries specifying ‘pipe’ will employ popen, their ‘.fmts’ element could contain many pipelined commands and, none can be interactive.

If the file or pipeline represented in your ‘.fmts’ deals with the specific PID input or accepted when prompted, then the format string must also contain the ‘%d’ specifier, as these examples illustrate.

```
.fmts= /proc/%d/numa_maps
(fmts= lsof -P -p%d
```

For ‘pipe’ type entries only, you may also wish to redirect stderr to stdout for a more comprehensive result. Thus the format string becomes:

```
.fmts= pmap -x %d2>&1
```

Here are examples of both types of Inspect entries as they might appear in the rcf file. The first entry will be ignored due to the initial ‘#’ character. For clarity, the pseudo tab depictions (‘I’) are surrounded by an extra space but the actual tabs would not be.

```
# pipe ^I Sockets ^I lsof -n -P -i 2>&1
pipe ^I Open Files ^I lsof -P -p %d 2>&1
file ^I NUMA Info ^I /proc/%d/numa_maps
pipe ^I Log ^I tail -n100 /var/log/syslog | sort -Mr
```

Except for the commented entry above, these next examples show what could be echoed to achieve similar results, assuming the rcf file name was ‘.toprc’. However, due to the embedded tab characters, each of these lines should be preceded by ‘**/bin/echo -e**’, not just a simple an ‘echo’, to enable backslash interpretation regardless of which shell you use.

```
"pipe\tOpen Files\tlsof -P -p %d 2>&1" >> ~/.toprc
"file\tNUMA Info\t/proc/%d/numa_maps" >> ~/.toprc
```

```
"pipe\|Log\|tail -n200 /var/log/syslog | sort -Mr" >> ~/.toprc
```

If any inspect entry you create produces output with unprintable characters they will be displayed in either the ^C notation or hexadecimal <FF> form, depending on their value. This applies to tab characters as well, which will show as '^I'. If you want a truer representation, any embedded tabs should be expanded. The following example takes what could have been a 'file' entry but employs a 'pipe' instead so as to expand the embedded tabs.

```
# next would have contained '\t' ...
# file ^I <your_name> ^I /proc/%d/status
# but this will eliminate embedded '\t' ...
pipe ^I <your_name> ^I cat /proc/%d/status | expand -
```

Note: Some programs might rely on *SIGINT* to end. Therefore, if a '**pipe**' such as the following is established, one must use Ctrl-C to terminate it in order to review the results. This is the single occasion where a '^C' will not also terminate top.

```
pipe ^I Trace ^I /usr/bin/strace -p %d 2>&1
```

Lastly, while '**pipe**' type entries have been discussed in terms of pipelines and commands, there is nothing to prevent you from including *shell scripts* as well. Perhaps even newly created scripts designed specifically for the 'Y' interactive command.

For example, as the number of your Inspect entries grows over time, the 'Options:' row will be truncated when screen width is exceeded. That does not affect operation other than to make some selections invisible. However, if some choices are lost to truncation but you want to see more options, there is an easy solution hinted at below.

```
Inspection Pause at pid ...
Use: left/right then <Enter> ...
Options: help 1 2 3 4 5 6 7 8 9 10 11 ...
```

The entries in the top rconfig file would have a number for the '.name' element and the 'help' entry would identify a shell script you've written explaining what those numbered selections actually mean. In that way, many more choices can be made visible.

6c. SYSTEM Configuration File

This configuration file represents defaults for users who have not saved their own configuration file. The format mirrors exactly the personal configuration file and can also include 'inspect' entries as explained above.

Creating it is a simple process.

1. Configure top appropriately for your installation and preserve that configuration with the 'W' interactive command.
2. Add and test any desired 'inspect' entries.
3. Copy that configuration file to the */etc/* directory as '**topdefaultrc**'.

6d. SYSTEM Restrictions File

The presence of this file will influence which version of the help screen is shown to an ordinary user.

More importantly, it will limit what ordinary users are allowed to do when top is running. They will not be

able to issue the following commands.

- k Kill a task
- r Renice a task
- d or s Change delay/sleep interval

This configuration file is not created by top. Rather, it is created manually and placed it in the */etc/* directory as ‘**toprc**’.

It should have exactly two lines, as shown in this example:

- s # line 1: secure mode switch
- 5.0 # line 2: delay interval in seconds

7. STUPID TRICKS Sampler

Many of these tricks work best when you give top a scheduling boost. So plan on starting him with a nice value of **-10**, assuming you’ve got the authority.

7a. Kernel Magic

For these stupid tricks, top needs full-screen mode.

- The user interface, through prompts and help, intentionally implies that the delay interval is limited to tenths of a second. However, you’re free to set any desired delay. If you want to see Linux at his scheduling best, try a delay of .09 seconds or less.

For this experiment, under x-windows open an xterm and maximize it. Then do the following:

- . provide a scheduling boost and tiny delay via:
nice -n -10 top -d.09
- . keep sorted column highlighting *Off* so as to minimize path length
- . turn *On* reverse row highlighting for emphasis
- . try various sort columns (TIME/MEM work well), and normal or reverse sorts to bring the most active processes into view

What you’ll see is a very busy Linux doing what he’s always done for you, but there was no program available to illustrate this.

- Under an xterm using ‘white-on-black’ colors, on top’s Color Mapping screen set the task color to black and be sure that task highlighting is set to bold, not reverse. Then set the delay interval to around .3 seconds.

After bringing the most active processes into view, what you’ll see are the ghostly images of just the currently running tasks.

- Delete the existing rcfile, or create a new symlink. Start this new version then type ‘T’ (a secret key, see topic 4c. Task Area Commands, SORTING) followed by ‘W’ and ‘q’. Finally, restart the program with **-d0** (zero delay).

Your display will be refreshed at three times the rate of the former top, a 300% speed advantage. As top climbs the TIME ladder, be as patient as you can while speculating on whether or not top will ever reach the top.

7b. Bouncing Windows

For these stupid tricks, top needs alternate–display mode.

- With 3 or 4 task displays visible, pick any window other than the last and turn idle processes *Off* using the ‘i’ command toggle. Depending on where you applied ‘i’, sometimes several task displays are bouncing and sometimes it’s like an accordion, as top tries his best to allocate space.
- Set each window’s summary lines differently: one with no memory (‘m’); another with no states (‘t’); maybe one with nothing at all, just the message line. Then hold down ‘a’ or ‘w’ and watch a variation on bouncing windows — hopping windows.
- Display all 4 windows and for each, in turn, set idle processes to *Off* using the ‘i’ command toggle. You’ve just entered the “extreme bounce” zone.

7c. The Big Bird Window

This stupid trick also requires alternate–display mode.

- Display all 4 windows and make sure that 1:Def is the ‘current’ window. Then, keep increasing window size with the ‘n’ interactive command until all the other task displays are “pushed out of the nest”.

When they’ve all been displaced, toggle between all visible/invisible windows using the ‘_’ command toggle. Then ponder this:

is top fibbing or telling honestly your imposed truth?

7d. The Ol’ Switcheroo

This stupid trick works best without alternate–display mode, since justification is active on a per window basis.

- Start top and make COMMAND the last (rightmost) column displayed. If necessary, use the ‘c’ command toggle to display command lines and ensure that forest view mode is active with the ‘V’ command toggle.

Then use the up/down arrow keys to position the display so that some truncated command lines are shown (‘+’ in last position). You may have to resize your xterm to produce truncation.

Lastly, use the ‘j’ command toggle to make the COMMAND column right justified.

Now use the right arrow key to reach the COMMAND column. Continuing with the right arrow key, watch closely the direction of travel for the command lines being shown.

some lines travel left, while others travel right

eventually all lines will Switcheroo, and move right

8. BUGS

Please send bug reports to [<procps@freelists.org>](mailto:procps@freelists.org).

9. SEE Also

free(1), ps(1), uptime(1), atop(1), slabtop(1), vmstat(8), w(1)

NAME

touch – change file timestamps

SYNOPSIS

touch [*OPTION*]... *FILE*...

DESCRIPTION

Update the access and modification times of each *FILE* to the current time.

A *FILE* argument that does not exist is created empty, unless **-c** or **-h** is supplied.

A *FILE* argument string of **-** is handled specially and causes touch to change the times of the file associated with standard output.

Mandatory arguments to long options are mandatory for short options too.

-a change only the access time

-c, --no-create
do not create any files

-d, --date=STRING
parse STRING and use it instead of current time

-f (ignored)

-h, --no-dereference

affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps of a symlink)

-m change only the modification time

-r, --reference=FILE
use this file's times instead of current time

-t STAMP
use [[CC]YY]MMDDhhmm[.ss] instead of current time

--time=WORD
change the specified time: WORD is access, atime, or use: equivalent to **-a** WORD is modify or mtime: equivalent to **-m**

--help display this help and exit

--version
output version information and exit

Note that the **-d** and **-t** options accept different time–date formats.

DATE STRING

The **--date=STRING** is a mostly free format human readable date string such as "Sun, 29 Feb 2004 16:21:42 -0800" or "2004-02-29 16:21:42" or even "next Thursday". A date string may contain items indicating calendar date, time of day, time zone, day of week, relative time, relative date, and numbers. An empty string indicates the beginning of the day. The date string format is more complex than is easily documented here but is fully described in the info documentation.

AUTHOR

Written by Paul Rubin, Arnold Robbins, Jim Kingdon, David MacKenzie, and Randy Smith.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent

permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/touch>>
or available locally via: info '(coreutils) touch invocation'

NAME

tr – translate or delete characters

SYNOPSIS

tr [*OPTION*]... *SET1* [*SET2*]

DESCRIPTION

Translate, squeeze, and/or delete characters from standard input, writing to standard output.

-c, --complement

use the complement of SET1

-d, --delete

delete characters in SET1, do not translate

-s, --squeeze-repeats

replace each sequence of a repeated character that is listed in the last specified SET, with a single occurrence of that character

-t, --truncate-set1

first truncate SET1 to length of SET2

--help display this help and exit

--version

output version information and exit

SETS are specified as strings of characters. Most represent themselves. Interpreted sequences are:

\NNN character with octal value NNN (1 to 3 octal digits)

\\\ backslash

\a audible BEL

\b backspace

\f form feed

\n new line

\r return

\t horizontal tab

\v vertical tab

CHAR1–CHAR2

all characters from CHAR1 to CHAR2 in ascending order

[CHAR*]

in SET2, copies of CHAR until length of SET1

[CHAR*REPEAT]

REPEAT copies of CHAR, REPEAT octal if starting with 0

:alnum:]

all letters and digits

:alpha:]

all letters

:blank:]

all horizontal whitespace

:cntrl:] all control characters

:digit:] all digits

[**:graph:**] all printable characters, not including space
[**:lower:**] all lower case letters
[**:print:**] all printable characters, including space
[**:punct:**] all punctuation characters
[**:space:**] all horizontal or vertical whitespace
[**:upper:**] all upper case letters
[**:xdigit:**] all hexadecimal digits
[=CHAR=] all characters which are equivalent to CHAR

Translation occurs if **-d** is not given and both SET1 and SET2 appear. **-t** may be used only when translating. SET2 is extended to length of SET1 by repeating its last character as necessary. Excess characters of SET2 are ignored. Only [**:lower:**] and [**:upper:**] are guaranteed to expand in ascending order; used in SET2 while translating, they may only be used in pairs to specify case conversion. **-s** uses the last specified SET, and occurs after translation or deletion.

AUTHOR

Written by Jim Meyering.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/tr>>
or available locally via: info '(coreutils) tr invocation'

NAME

tty – print the file name of the terminal connected to standard input

SYNOPSIS

tty [*OPTION*]...

DESCRIPTION

Print the file name of the terminal connected to standard input.

-s, --silent, --quiet

print nothing, only return an exit status

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/tty>>

or available locally via: info '(coreutils) tty invocation'

NAME

tty – controlling terminal

DESCRIPTION

The file */dev/tty* is a character file with major number 5 and minor number 0, usually with mode 0666 and ownership root:tty. It is a synonym for the controlling terminal of a process, if any.

In addition to the **ioctl**(2) requests supported by the device that **tty** refers to, the **ioctl**(2) request **TIOCNOTTY** is supported.

TIOCNOTTY

Detach the calling process from its controlling terminal.

If the process is the session leader, then **SIGHUP** and **SIGCONT** signals are sent to the foreground process group and all processes in the current session lose their controlling tty.

This **ioctl**(2) call works only on file descriptors connected to */dev/tty*. It is used by daemon processes when they are invoked by a user at a terminal. The process attempts to open */dev/tty*. If the open succeeds, it detaches itself from the terminal by using **TIOCNOTTY**, while if the open fails, it is obviously not attached to a terminal and does not need to detach itself.

FILES

/dev/tty

SEE ALSO

chown(1), **mknod**(1), **ioctl**(2), **ioctl_console**(2), **ioctl_tty**(2), **termios**(3), **ttyS**(4), **vcs**(4), **pty**(7), **agetty**(8), **mingetty**(8)

NAME

ttyS – serial terminal lines

DESCRIPTION

ttyS[0–3] are character devices for the serial terminal lines.

They are typically created by:

```
mknod -m 660 /dev/ttys0 c 4 64 # base address 0x3f8
mknod -m 660 /dev/ttys1 c 4 65 # base address 0x2f8
mknod -m 660 /dev/ttys2 c 4 66 # base address 0x3e8
mknod -m 660 /dev/ttys3 c 4 67 # base address 0x2e8
chown root:tty /dev/ttys[0-3]
```

FILES

/dev/ttys[0–3]

SEE ALSO

chown(1), mknod(1), tty(4), agetty(8), mingetty(8), setserial(8)

NAME

ioctl_tty – ioctls for terminals and serial lines

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/ioctl.h>
#include <asm/termbits.h> /* Definition of struct termios,
                           struct termios2, and
                           Bnnn, BOTHER, CBAUD, CLOCAL,
                           TC*{FLUSH,ON,OFF} and other constants */

int ioctl(int fd, int cmd, ...);
```

DESCRIPTION

The **ioctl**(2) call for terminals and serial ports accepts many possible command arguments. Most require a third argument, of varying type, here called *argp* or *arg*.

Use of **ioctl()** makes for nonportable programs. Use the POSIX interface described in **termios**(3) whenever possible.

Please note that **struct termios** from *<asm/termbits.h>* is different and incompatible with **struct termios** from *<termios.h>*. These ioctl calls require **struct termios** from *<asm/termbits.h>*.

Get and set terminal attributes**TCGETS**

Argument: **struct termios** **argp*

Equivalent to *tcgetattr(fd, argp)*.

Get the current serial port settings.

TCSETS

Argument: **const struct termios** **argp*

Equivalent to *tcsetattr(fd, TCSANOW, argp)*.

Set the current serial port settings.

TCSETSW

Argument: **const struct termios** **argp*

Equivalent to *tcsetattr(fd, TCSADRAIN, argp)*.

Allow the output buffer to drain, and set the current serial port settings.

TCSETSF

Argument: **const struct termios** **argp*

Equivalent to *tcsetattr(fd, TCSAFLUSH, argp)*.

Allow the output buffer to drain, discard pending input, and set the current serial port settings.

The following four ioctls, added in Linux 2.6.20, are just like **TCGETS**, **TCSETS**, **TCSETSW**, **TCSETSF**, except that they take a *struct termios2* * instead of a *struct termios* *. If the structure member **c_cflag** contains the flag **BOTHER**, then the baud rate is stored in the structure members **c_ispeed** and **c_ospeed** as integer values. These ioctls are not supported on all architectures.

TCGETS2	struct termios2 * <i>argp</i>
TCSETS2	const struct termios2 * <i>argp</i>
TCSETSW2	const struct termios2 * <i>argp</i>
TCSETSF2	const struct termios2 * <i>argp</i>

The following four ioctls are just like **TCGETS**, **TCSETS**, **TCSETSW**, **TCSETSF**, except that they take a *struct termio* * instead of a *struct termios* *.

TCGETA	struct termio *argp
TCSETA	const struct termio *argp
TCSETAW	const struct termio *argp
TCSETAF	const struct termio *argp

Locking the termios structure

The *termios* structure of a terminal can be locked. The lock is itself a *termios* structure, with nonzero bits or fields indicating a locked value.

TIOCGLKTRMIOS

Argument: **struct termios *argp**

Gets the locking status of the *termios* structure of the terminal.

TIOCSLKTRMIOS

Argument: **const struct termios *argp**

Sets the locking status of the *termios* structure of the terminal. Only a process with the **CAP_SYS_ADMIN** capability can do this.

Get and set window size

Window sizes are kept in the kernel, but not used by the kernel (except in the case of virtual consoles, where the kernel will update the window size when the size of the virtual console changes, for example, by loading a new font).

TIOCGWINSZ

Argument: **struct winsize *argp**

Get window size.

TIOCSWINSZ

Argument: **const struct winsize *argp**

Set window size.

The struct used by these ioctls is defined as

```
struct winsize {
    unsigned short ws_row;
    unsigned short ws_col;
    unsigned short ws_xpixel; /* unused */
    unsigned short ws_ypixel; /* unused */
};
```

When the window size changes, a **SIGWINCH** signal is sent to the foreground process group.

Sending a break**TCSBRK**

Argument: **int arg**

Equivalent to *tcsendbreak(fd, arg)*.

If the terminal is using asynchronous serial data transmission, and *arg* is zero, then send a break (a stream of zero bits) for between 0.25 and 0.5 seconds. If the terminal is not using asynchronous serial data transmission, then either a break is sent, or the function returns without doing anything. When *arg* is nonzero, nobody knows what will happen.

(SVr4, UnixWare, Solaris, and Linux treat *tcsendbreak(fd,arg)* with nonzero *arg* like *tcdrain(fd)*. SunOS treats *arg* as a multiplier, and sends a stream of bits *arg* times as long as done for zero *arg*. DG/UX and AIX treat *arg* (when nonzero) as a time interval measured in milliseconds. HP-UX ignores *arg*.)

TCSBRKP

Argument: **int arg**

So-called "POSIX version" of **TCSBRK**. It treats nonzero *arg* as a time interval measured in microseconds, and does nothing when the driver does not support breaks.

TIOCSBRK

Argument: **void**

Turn break on, that is, start sending zero bits.

TIOCCBRK

Argument: **void**

Turn break off, that is, stop sending zero bits.

Software flow control

TCXONC

Argument: **int arg**

Equivalent to *tcflow(fd, arg)*.

See **tcflow(3)** for the argument values **TCOOFF**, **TCOON**, **TCIOFF**, **TCION**.

Buffer count and flushing

FIONREAD

Argument: **int *argp**

Get the number of bytes in the input buffer.

TIOCINQ

Argument: **int *argp**

Same as **FIONREAD**.

TIOCOUTQ

Argument: **int *argp**

Get the number of bytes in the output buffer.

TCFLSH

Argument: **int arg**

Equivalent to *tcflush(fd, arg)*.

See **tcflush(3)** for the argument values **TCIFLUSH**, **TCOFLUSH**, **TCIOFLUSH**.

TIOCSERGETLSR

Argument: **int *argp**

Get line status register. Status register has **TIOCSER_TEMT** bit set when output buffer is empty and also hardware transmitter is physically empty.

Does not have to be supported by all serial tty drivers.

tcdrain(3) does not wait and returns immediately when **TIOCSER_TEMT** bit is set.

Faking input

TIOCSTI

Argument: **const char *argp**

Insert the given byte in the input queue.

Redirecting console output

TIOCCONS

Argument: **void**

Redirect output that would have gone to */dev/console* or */dev/tty0* to the given terminal. If that was a pseudoterminal master, send it to the slave. Before Linux 2.6.10, anybody can do this as long as the output was not redirected yet; since Linux 2.6.10, only a process with the **CAP_SYS_ADMIN** capability may do this. If output was redirected already, then **EBUSY** is returned, but redirection can be stopped by using this ioctl with *fd* pointing at */dev/console* or

/dev/tty0.

Controlling terminal

TIOCSCTTY

Argument: **int** *arg*

Make the given terminal the controlling terminal of the calling process. The calling process must be a session leader and not have a controlling terminal already. For this case, *arg* should be specified as zero.

If this terminal is already the controlling terminal of a different session group, then the ioctl fails with **EPERM**, unless the caller has the **CAP_SYS_ADMIN** capability and *arg* equals 1, in which case the terminal is stolen, and all processes that had it as controlling terminal lose it.

TIOCNOTTY

Argument: **void**

If the given terminal was the controlling terminal of the calling process, give up this controlling terminal. If the process was session leader, then send **SIGHUP** and **SIGCONT** to the foreground process group and all processes in the current session lose their controlling terminal.

Process group and session ID

TIOCGPGRP

Argument: **pid_t** **argp*

When successful, equivalent to **argp = tcgetpgrp(fd)*.

Get the process group ID of the foreground process group on this terminal.

TIOCSPGRP

Argument: **const pid_t** **argp*

Equivalent to *tcsetpgrp(fd, *argp)*.

Set the foreground process group ID of this terminal.

TIOCGSID

Argument: **pid_t** **argp*

When successful, equivalent to **argp = tcgetsid(fd)*.

Get the session ID of the given terminal. This fails with the error **ENOTTY** if the terminal is not a master pseudoterminal and not our controlling terminal. Strange.

Exclusive mode

TIOCEXCL

Argument: **void**

Put the terminal into exclusive mode. No further **open(2)** operations on the terminal are permitted. (They fail with **EBUSY**, except for a process with the **CAP_SYS_ADMIN** capability.)

TIOCGEXCL

Argument: **int** **argp*

(since Linux 3.8) If the terminal is currently in exclusive mode, place a nonzero value in the location pointed to by *argp*; otherwise, place zero in **argp*.

TIOCNXCL

Argument: **void**

Disable exclusive mode.

Line discipline

TIOCGETD

Argument: **int** **argp*

Get the line discipline of the terminal.

TIOCSETD

Argument: **const int *argp**

Set the line discipline of the terminal.

Pseudoterminal ioctls**TIOCPKT**

Argument: **const int *argp**

Enable (when **argp* is nonzero) or disable packet mode. Can be applied to the master side of a pseudoterminal only (and will return **ENOTTY** otherwise). In packet mode, each subsequent **read(2)** will return a packet that either contains a single nonzero control byte, or has a single byte containing zero ('\0') followed by data written on the slave side of the pseudoterminal. If the first byte is not **TIOCPKT_DATA** (0), it is an OR of one or more of the following bits:

TIOCPKT_FLUSHREAD	The read queue for the terminal is flushed.
TIOCPKT_FLUSHWRITE	The write queue for the terminal is flushed.
TIOCPKT_STOP	Output to the terminal is stopped.
TIOCPKT_START	Output to the terminal is restarted.
TIOCPKT_DOSTOP	The start and stop characters are ^S/^Q .
TIOCPKT_NOSTOP	The start and stop characters are not ^S/^Q .

While packet mode is in use, the presence of control status information to be read from the master side may be detected by a **select(2)** for exceptional conditions or a **poll(2)** for the **POLLPRI** event.

This mode is used by **rlogin(1)** and **rlogind(8)** to implement a remote-echoed, locally **^S/^Q** flow-controlled remote login.

TIOCGPkt

Argument: **const int *argp**

(since Linux 3.8) Return the current packet mode setting in the integer pointed to by *argp*.

TIOCSPTLCK

Argument: **int *argp**

Set (if **argp* is nonzero) or remove (if **argp* is zero) the lock on the pseudoterminal slave device. (See also **unlockpt(3)**.)

TIOCGPTLCK

Argument: **int *argp**

(since Linux 3.8) Place the current lock state of the pseudoterminal slave device in the location pointed to by *argp*.

TIOCGPTPEER

Argument: **int flags**

(since Linux 4.13) Given a file descriptor in *fd* that refers to a pseudoterminal master, open (with the given **open(2)**-style *flags*) and return a new file descriptor that refers to the peer pseudoterminal slave device. This operation can be performed regardless of whether the pathname of the slave device is accessible through the calling process's mount namespace.

Security-conscious programs interacting with namespaces may wish to use this operation rather than **open(2)** with the pathname returned by **ptsname(3)**, and similar library functions that have insecure APIs. (For example, confusion can occur in some cases using **ptsname(3)** with a pathname where a devpts filesystem has been mounted in a different mount namespace.)

The BSD ioctls **TIOCSTOP**, **TIOCSTART**, **TIOCUCNTL**, and **TIOCREMOTE** have not been

implemented under Linux.

Modem control

TIOCMGET

Argument: **int *argp**

Get the status of modem bits.

TIOCMSET

Argument: **const int *argp**

Set the status of modem bits.

TIOCMBIC

Argument: **const int *argp**

Clear the indicated modem bits.

TIOCMBIS

Argument: **const int *argp**

Set the indicated modem bits.

The following bits are used by the above ioctl:

TIOCM_LE	DSR (data set ready/line enable)
TIOCM_DTR	DTR (data terminal ready)
TIOCM_RTS	RTS (request to send)
TIOCM_ST	Secondary TxD (transmit)
TIOCM_SR	Secondary RxD (receive)
TIOCM_CTS	CTS (clear to send)
TIOCM_CAR	DCD (data carrier detect)
TIOCM_CD	see TIOCM_CAR
TIOCM RNG	RNG (ring)
TIOCM RI	see TIOCM RNG
TIOCM_DSR	DSR (data set ready)

TIOCMIWAIT

Argument: **int arg**

Wait for any of the 4 modem bits (DCD, RI, DSR, CTS) to change. The bits of interest are specified as a bit mask in *arg*, by ORing together any of the bit values, **TIOCM RNG**, **TIOCM_DSR**, **TIOCM_CD**, and **TIOCM_CTS**. The caller should use **TIOCGCOUNT** to see which bit has changed.

TIOCGCOUNT

Argument: **struct serial_icounter_struct *argp**

Get counts of input serial line interrupts (DCD, RI, DSR, CTS). The counts are written to the *serial_icounter_struct* structure pointed to by *argp*.

Note: both 1->0 and 0->1 transitions are counted, except for RI, where only 0->1 transitions are counted.

Marking a line as local

TIOCGSOFTCAR

Argument: **int *argp**

("Get software carrier flag") Get the status of the CLOCAL flag in the c_cflag field of the *termios* structure.

TIOCSSOFTCAR

Argument: **const int *argp**

("Set software carrier flag") Set the CLOCAL flag in the *termios* structure when **argp* is nonzero, and clear it otherwise.

If the **CLOCAL** flag for a line is off, the hardware carrier detect (DCD) signal is significant, and an **open(2)** of the corresponding terminal will block until DCD is asserted, unless the **O_NONBLOCK** flag is given. If **CLOCAL** is set, the line behaves as if DCD is always asserted. The software carrier flag is usually turned on for local devices, and is off for lines with modems.

Linux-specific

For the **TIOCLINUX** ioctl, see **ioctl_console(2)**.

Kernel debugging

```
#include <linux/tty.h>
```

```
TIOCTTYGSTRUCT
```

Argument: **struct tty_struct *argp**

Get the *tty_struct* corresponding to *fd*. This command was removed in Linux 2.5.67.

RETURN VALUE

The **ioctl(2)** system call returns 0 on success. On error, it returns -1 and sets *errno* to indicate the error.

ERRORS

EINVAL

Invalid command parameter.

ENOIOCTLCMD

Unknown command.

ENOTTY

Inappropriate *fd*.

EPERM

Insufficient permission.

EXAMPLES

Check the condition of DTR on the serial port.

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <unistd.h>

int
main(void)
{
    int fd, serial;

    fd = open("/dev/ttyS0", O_RDONLY);
    ioctl(fd, TIOCMGET, &serial);
    if (serial & TIOCM_DTR)
        puts("TIOCM_DTR is set");
    else
        puts("TIOCM_DTR is not set");
    close(fd);
}
```

Get or set arbitrary baudrate on the serial port.

```
/* SPDX-License-Identifier: GPL-2.0-or-later */

#include <asm/termbits.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
```

```

#include <sys/	ioctl.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
#if !defined BOTHER
    fprintf(stderr, "BOTHER is unsupported\n");
    /* Program may fallback to TCGETS/TCSETS with Bnnn constants */
    exit(EXIT_FAILURE);
#else
    /* Declare tio structure, its type depends on supported ioctl */
# if defined TCGETS2
    struct termios2 tio;
# else
    struct termios tio;
# endif
    int fd, rc;

    if (argc != 2 && argc != 3 && argc != 4) {
        fprintf(stderr, "Usage: %s device [output [input] ]\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    fd = open(argv[1], O_RDWR | O_NONBLOCK | O_NOCTTY);
    if (fd < 0) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    /* Get the current serial port settings via supported ioctl */
# if defined TCGETS2
    rc = ioctl(fd, TCGETS2, &tio);
# else
    rc = ioctl(fd, TCGETS, &tio);
# endif
    if (rc) {
        perror("TCGETS");
        close(fd);
        exit(EXIT_FAILURE);
    }

    /* Change baud rate when more arguments were provided */
    if (argc == 3 || argc == 4) {
        /* Clear the current output baud rate and fill a new value */
        tio.c_cflag &= ~CBAUD;
        tio.c_cflag |= BOTHER;
        tio.c_ospeed = atoi(argv[2]);

        /* Clear the current input baud rate and fill a new value */
        tio.c_cflag &= ~(CBAUD << IBSHIFT);
        tio.c_cflag |= BOTHER << IBSHIFT;
        /* When 4th argument is not provided reuse output baud rate */
        tio.c_ispeed = (argc == 4) ? atoi(argv[3]) : atoi(argv[2]);
    }
}

```

```
/* Set new serial port settings via supported ioctl */
# if defined TCSETS2
    rc = ioctl(fd, TCSETS2, &tio);
# else
    rc = ioctl(fd, TCSETS, &tio);
# endif
    if (rc) {
        perror("TCSETS");
        close(fd);
        exit(EXIT_FAILURE);
    }

    /* And get new values which were really configured */
# if defined TCGETS2
    rc = ioctl(fd, TCGETS2, &tio);
# else
    rc = ioctl(fd, TCGETS, &tio);
# endif
    if (rc) {
        perror("TCGETS");
        close(fd);
        exit(EXIT_FAILURE);
    }
}

close(fd);

printf("output baud rate: %u\n", tio.c_ospeed);
printf("input baud rate: %u\n", tio.c_ispeed);

exit(EXIT_SUCCESS);
#endif
}
```

SEE ALSO

lattach(8), ioctl(2), ioctl_console(2), termios(3), pty(7)

NAME

ufw—framework – using the ufw framework

DESCRIPTION

ufw provides both a command line interface and a framework for managing a netfilter firewall. While the **ufw** command provides an easy to use interface for managing a firewall, the **ufw** framework provides the administrator methods to customize default behavior and add rules not supported by the command line tool. In this way, **ufw** can take full advantage of Linux netfilter's power and flexibility.

OVERVIEW

The framework provides boot time initialization, rules files for adding custom rules, a method for loading netfilter modules, configuration of kernel parameters and configuration of IPv6. The framework consists of the following files:

```
/lib/ufw/ufw-init
    initialization script

/etc/ufw(before).init
    initialization customization script run before ufw is initialized

/etc/ufw(after).init
    initialization customization script run after ufw is initialized

/etc/ufw(before[6].rules
    rules file containing rules evaluated before UI added rules

/etc/ufw(user[6].rules
    rules file containing UI added rules (managed with the ufw command)

/etc/ufw(after[6].rules
    rules file containing rules evaluated after UI added rules

/etc/default/ufw
    high level configuration

/etc/ufw/sysctl.conf
    kernel network tunables

/etc/ufw/ufw.conf
    additional high level configuration
```

BOOT INITIALIZATION

ufw is started on boot with /lib/ufw/ufw-init. This script is a standard SysV style initscript used by the **ufw** command and should not be modified. The /etc/before.init and /etc/after.init scripts may be used to perform any additional firewall configuration that is not yet supported in ufw itself and if they exist and are executable, ufw-init will execute these scripts. ufw-init will exit with error if either of these scripts exit with error. ufw-init supports the following arguments:

start:	loads the firewall
stop:	unloads the firewall
restart:	reloads the firewall
force-reload:	same as restart
status:	basic status of the firewall
force-stop:	same as stop, except does not check if the firewall is already loaded

flush-all:

flushes the built-in chains, deletes all non-built-in chains and resets the policy to ACCEPT
 ufw-init will call before.init and after.init with start, stop, status and flush-all, but typically, if used, these scripts need only implement start and stop.

ufw uses many user-defined chains in addition to the built-in iptables chains. If MANAGE_BUILTINS in /etc/default/ufw is set to 'yes', on stop and reload the built-in chains are flushed. If it is set to 'no', on stop and reload the **ufw** secondary chains are removed and the **ufw** primary chains are flushed. In addition to flushing the **ufw** specific chains, it keeps the primary chains in the same order with respect to any other user-defined chains that may have been added. This allows for **ufw** to interoperate with other software that may manage their own firewall rules.

To ensure your firewall is loading on boot, you must integrate this script into the boot process. Consult your distribution's documentation for the proper way to modify your boot process if **ufw** is not already integrated.

RULES FILES

ufw is in part a front-end for **iptables-restore**, with its rules saved in /etc/ufw/before.rules, /etc/ufw/after.rules and /etc/ufw/user.rules. Administrators can customize **before.rules** and **after.rules** as desired using the standard **iptables-restore** syntax. Rules are evaluated as follows: **before.rules** first, **user.rules** next, and **after.rules** last. IPv6 rules are evaluated in the same way, with the rules files named **before6.rules**, **user6.rules** and **after6.rules**. Please note that **ufw status** only shows rules added with **ufw** and not the rules found in the /etc/ufw rules files.

Important: **ufw** only uses the *filter table by default. You may add any other tables such as *nat, *raw and *mangle as desired. For each table a corresponding COMMIT statement is required.

After modifying any of these files, you must reload **ufw** for the rules to take effect. See the EXAMPLES section for common uses of these rules files.

MODULES

Netfilter has many different connection tracking modules. These modules are aware of the underlying protocol and allow the administrator to simplify his or her rule sets. You can adjust which netfilter modules to load by adjusting IPT_MODULES in /etc/default/ufw. Some popular modules to load are:

```
nf_conntrack_ftp
nf_nat_ftp
nf_conntrack_irc
nf_nat_irc
nf_conntrack_netbios_ns
nf_conntrack_pptp
nf_conntrack_tftp
nf_nat_tftp
nf_conntrack_sane
```

Unconditional loading of connection tracking modules (nf_conntrack_*) in this manner is deprecated. **ufw** continues to support the functionality but new configuration should only contain the specific modules required for the site. For more information, see CONNECTION HELPERS.

KERNEL PARAMETERS

ufw will read in /etc/ufw/sysctl.conf on boot when enabled. Please note that /etc/ufw/sysctl.conf overrides values in the system sysctl.conf (usually /etc/sysctl.conf). Administrators can change the file used by modifying /etc/default/ufw.

IPV6

IPv6 is enabled by default. When disabled, all incoming, outgoing and forwarded packets are dropped, with the exception of traffic on the loopback interface. To adjust this behavior, set IPV6 to 'yes' in /etc/default/ufw. See the **ufw** manual page for details.

EXAMPLES

As mentioned, **ufw** loads its rules files into the kernel by using the **iptables-restore** and **ip6tables-restore** commands. Users wanting to add rules to the **ufw** rules files manually must be familiar with these as well as the **iptables** and **ip6tables** commands. Below are some common examples of using the **ufw** rules files. All examples assume IPv4 only and that DEFAULT_FORWARD_POLICY in /etc/default/ufw is set to DROP.

IP Masquerading

To allow IP masquerading for computers from the 10.0.0.0/8 network on eth1 to share the single IP address on eth0:

Edit /etc/ufw/sysctl.conf to have:

```
net.ipv4.ip_forward=1
```

Add to the end of /etc/ufw/before.rules, after the *filter section:

```
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.0.0.0/8 -o eth0 -j MASQUERADE
COMMIT
```

If your firewall is using IPv6 tunnels or 6to4 and is also doing NAT, then you should not usually masquerade protocol '41' (ipv6) packets. For example, instead of the above, /etc/ufw/before.rules can be adjusted to have:

```
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.0.0.0/8 ! --protocol 41 -o eth0 -j MASQUERADE
COMMIT
```

Add the **ufw route** to allow the traffic:

```
ufw route allow in on eth1 out on eth0 from 10.0.0.0/8
```

Port Redirections

To forward tcp port 80 on eth0 to go to the webserver at 10.0.0.2:

Edit /etc/ufw/sysctl.conf to have:

```
net.ipv4.ip_forward=1
```

Add to the end of /etc/ufw/before.rules, after the *filter section:

```
*nat
:PREROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -i eth0 --dport 80 -j DNAT \
--to-destination 10.0.0.2:80
COMMIT
```

Add the **ufw route** rule to allow the traffic:

```
ufw route allow in on eth0 to 10.0.0.2 port 80 proto tcp
```

Egress filtering

To block RFC1918 addresses going out of eth0:

Add the **ufw route** rules to reject the traffic:

```
ufw route reject out on eth0 to 10.0.0.0/8
ufw route reject out on eth0 to 172.16.0.0/12
ufw route reject out on eth0 to 192.168.0.0/16
```

Full example

This example combines the other examples and demonstrates a simple routing firewall. **Warning:** this setup is only an example to demonstrate the functionality of the **ufw** framework in a concise and simple manner and should not be used in production without understanding what each part does and does not do. Your firewall will undoubtedly want to be less open.

This router/firewall has two interfaces: eth0 (Internet facing) and eth1 (internal LAN). Internal clients have addresses on the 10.0.0.0/8 network and should be able to connect to anywhere on the Internet. Connections to port 80 from the Internet should be forwarded to 10.0.0.2. Access to ssh port 22 from the administrative workstation (10.0.0.100) to this machine should be allowed. Also make sure no internal traffic goes to the Internet.

Edit /etc/ufw/sysctl.conf to have:

```
net.ipv4.ip_forward=1
```

Add to the end of /etc/ufw/before.rules, after the *filter section:

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -i eth0 --dport 80 -j DNAT \
--to-destination 10.0.0.2:80
-A POSTROUTING -s 10.0.0.0/8 -o eth0 -j MASQUERADE
COMMIT
```

Add the necessary **ufw** rules:

```
ufw route reject out on eth0 to 10.0.0.0/8
ufw route reject out on eth0 to 172.16.0.0/12
ufw route reject out on eth0 to 192.168.0.0/16
ufw route allow in on eth1 out on eth0 from 10.0.0.0/8
ufw route allow in on eth0 to 10.0.0.2 port 80 proto tcp
ufw allow in on eth1 from 10.0.0.100 to any port 22 proto tcp
```

CONNECTION HELPERS

Various protocols require the use of netfilter connection tracking helpers to group related packets into RELATED flows to make rulesets clearer and more precise. For example, with a couple of kernel modules and a couple of rules, a ruleset could simply allow a connection to FTP port 21, then the kernel would examine the traffic and mark the other FTP data packets as RELATED to the initial connection.

When the helpers were first introduced, one could only configure the modules as part of module load (eg, if your FTP server listened on a different port than 21, you'd have to load the nf_conntrack_ftp module specifying the correct port). Over time it was understood that unconditionally using connection helpers could lead to abuse, in part because some protocols allow user specified data that would allow traversing the firewall in undesired ways. As of kernel 4.7, automatic conntrack helper assignment (ie, handling packets for a given port and all IP addresses) is disabled (the old behavior can be restored by setting net/netfilter/nf_conntrack_helper=1 in /etc/ufw/sysctl.conf). Firewalls should now instead use the CT target to associate traffic with a particular helper and then set RELATED rules to use the helper. This allows sites to tailor the use of helpers and help avoid abuse.

In general, to use helpers securely, the following needs to happen:

1. net/netfilter/nf_conntrack_helper should be set to 0 (default)
2. create a rule for the start of a connection (eg for FTP, port 21)
3. create a helper rule to associate the helper with this connection
4. create a helper rule to associate a RELATED flow with this connection
5. if needed, add the corresponding nf_conntrack_* module to IPT_MODULES

6. optionally add the corresponding `nf_nat_*` module to `IPT_MODULES`

In general it is desirable to make connection helper rules as specific as possible and ensure anti-spoofing is correctly setup for your site to avoid security issues in your ruleset. For more information, see ANTI-SPOOFING, above, and <<https://home.regit.org/netfilter-en/secure-use-of-helpers/>>.

Currently helper rules must be managed in via the RULES FILES. A future version of **ufw** will introduce syntax for working with helper rules.

NOTES

When using ufw with libvirt and bridging, packets may be blocked. The libvirt team recommends that the following sysctl's be set to disable netfilter on the bridge:

```
net.bridge.bridge-nf-call-ip6tables = 0  
net.bridge.bridge-nf-call-iptables = 0  
net.bridge.bridge-nf-call-arptables = 0
```

Note that the bridge module must be loaded in to the kernel before these values are set. One way to ensure this works properly with ufw is to add 'bridge' to `IPT_MODULES` in `/etc/default/ufw`, and then add the above rules to `/etc/ufw/sysctl.conf`.

Alternatively to disabling netfilter on the bridge, you can configure iptables to allow all traffic to be forwarded across the bridge. Eg, add to `/etc/ufw/before.rules` within the `*filter` section:

```
-I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
```

SEE ALSO

ufw(8), **iptables(8)**, **ip6tables(8)**, **iptables-restore(8)**, **ip6tables-restore(8)**, **sysctl(8)**, **sysctl.conf(5)**

AUTHOR

ufw is Copyright 2008-2021, Canonical Ltd.

NAME

ufw – program for managing a netfilter firewall

DESCRIPTION

This program is for managing a Linux firewall and aims to provide an easy to use interface for the user.

USAGE

```
ufw [--dry-run] enable|disable|reload
ufw [--dry-run] default allow|deny|reject [incoming|outgoing|routed]
ufw [--dry-run] logging on|off|LEVEL
ufw [--dry-run] reset
ufw [--dry-run] status [verbose|numbered]
ufw [--dry-run] show REPORT
ufw [--dry-run] [delete] [insert NUM] [prepend] allow|deny|reject|limit [in|out] [log|log-all] [
PORT/[PROTOCOL] | APPNAME ] [comment COMMENT]
ufw [--dry-run] [rule] [delete] [insert NUM] [prepend] allow|deny|reject|limit [in|out] [on INTERFACE] [log|log-all] [proto PROTOCOL] [from ADDRESS [port PORT | app APPNAME ]] [to ADDRESS [port PORT | app APPNAME ]] [comment COMMENT]
ufw [--dry-run] route [delete] [insert NUM] [prepend] allow|deny|reject|limit [in|out on INTERFACE] [log|log-all] [proto PROTOCOL] [from ADDRESS [port PORT | app APPNAME]] [to ADDRESS [port PORT | app APPNAME]] [comment COMMENT]
ufw [--dry-run] [--force] delete NUM
ufw [--dry-run] app list|info|default|update
```

OPTIONS**--version**

show program's version number and exit

-h, --help

show help message and exit

--dry-run

don't modify anything, just show the changes

enable reloads firewall and enables firewall on boot.

disable unloads firewall and disables firewall on boot

reload reloads firewall

default allow|deny|reject DIRECTION

change the default policy for traffic going DIRECTION, where DIRECTION is one of **incoming**, **outgoing** or **routed**. Note that existing rules will have to be migrated manually when changing the default policy. See **RULE SYNTAX** for more on **deny** and **reject**.

logging on|off|LEVEL

toggle logging. Logged packets use the LOG_KERN syslog facility. Systems configured for rsyslog support may also log to /var/log/ufw.log. Specifying a LEVEL turns logging on for the specified LEVEL. The default log level is 'low'. See **LOGGING** for details.

reset Disables and resets firewall to installation defaults. Can also give the **--force** option to perform the reset without confirmation.

status show status of firewall and ufw managed rules. Use **status verbose** for extra information. In the status output, 'Anywhere' is synonymous with 'any', 0.0.0.0/0 (IPv4) and ::/0 (IPv6). Note that when using **status**, there is a subtle difference when reporting interfaces. For example, if the

following rules are added:

```
ufw allow in on eth0 from 192.168.0.0/16
ufw allow out on eth1 to 10.0.0.0/8
ufw route allow in on eth0 out on eth1 to 10.0.0.0/8 from 192.168.0.0/16
ufw limit 2222/tcp comment 'SSH port'
```

ufw status will output:

To	Action	From
--	---	---
Anywhere on eth0	ALLOW	192.168.0.0/16
10.0.0.0/8	ALLOW OUT	Anywhere on eth1
10.0.0.0/8 on eth1	ALLOW FWD	192.168.0.0/16 on eth0
Anywhere	LIMIT	Anywhere # SSH port

For the input and output rules, the interface is reported relative to the firewall system as an endpoint, whereas with route rules, the interface is reported relative to the direction packets flow through the firewall.

show REPORT

display information about the running firewall. See **REPORTS**

allow ARGS

add allow rule. See **RULE SYNTAX**

deny ARGS

add deny rule. See **RULE SYNTAX**

reject ARGS

add reject rule. See **RULE SYNTAX**

limit ARGS

add limit rule. See **RULE SYNTAX**

delete RULE|NUM

deletes the corresponding RULE

insert NUM RULE

insert the corresponding RULE as rule number NUM

prepend RULE

prepend the corresponding RULE to the top of the ruleset

RULE SYNTAX

Users can specify rules using either a simple syntax or a full syntax. The simple syntax only specifies the port and optionally the protocol to be allowed or denied on the host.

Both syntaxes support specifying a comment for the rule. For existing rules, specifying a different comment updates the comment and specifying "" removes the comment.

Example rules using the simple syntax:

```
ufw allow 53
```

This rule will allow tcp and udp port 53 to any address on this host. To specify a protocol, append '/protocol' to the port. For example:

```
ufw allow 25/tcp
```

This will allow tcp port 25 to any address on this host. **ufw** will also check /etc/services for the port and protocol if specifying a service by name. Eg:

```
ufw allow smtp
```

ufw supports both ingress and egress filtering and users may optionally specify a direction of either **in** or **out** for either incoming or outgoing traffic. If no direction is supplied, the rule applies to incoming traffic. Eg:

```
ufw allow in http
ufw reject out smtp
ufw reject telnet comment 'telnet is unencrypted'
```

Users can also use a fuller syntax, specifying the source and destination addresses and ports. This syntax is loosely based on OpenBSD's PF syntax. For example:

```
ufw deny proto tcp to any port 80
```

This will deny all traffic to tcp port 80 on this host. Another example:

```
ufw deny proto tcp from 10.0.0.0/8 to 192.168.0.1 port 25
```

This will deny all traffic from the RFC1918 Class A network to tcp port 25 with the address 192.168.0.1.

```
ufw deny proto tcp from 2001:db8::/32 to any port 25
```

This will deny all traffic from the IPv6 2001:db8::/32 to tcp port 25 on this host. IPv6 must be enabled in /etc/default/ufw for IPv6 firewalling to work.

```
ufw deny in on eth0 to 224.0.0.1 proto igmp
```

This will deny all igmp traffic to 224.0.0.1 on the eth0 interface.

```
ufw allow in on eth0 to 192.168.0.1 proto gre
```

This will allow all gre traffic to 192.168.0.1 on the eth0 interface.

```
ufw allow proto tcp from any to any port 80,443,8080:8090 comment 'web app'
```

The above will allow all traffic to tcp ports 80, 443 and 8080–8090 inclusive and adds a comment for the rule. When specifying multiple ports, the ports list must be numeric, cannot contain spaces and must be modified as a whole. Eg, in the above example you cannot later try to delete just the '443' port. You cannot specify more than 15 ports (ranges count as 2 ports, so the port count in the above example is 4).

ufw supports several different protocols. The following are valid in any rule and enabled when the protocol is not specified:

```
tcp
udp
```

The following have certain restrictions and are not enabled when the protocol is not specified:

```
ah    valid without port number
```

```
esp  valid without port number
gre  valid without port number
ipv6 valid for IPv4 addresses and without port number
igmp valid for IPv4 addresses and without port number
```

Rules for traffic not destined for the host itself but instead for traffic that should be routed/forwarded through the firewall should specify the **route** keyword before the rule (routing rules differ significantly from PF syntax and instead take into account netfilter FORWARD chain conventions). For example:

```
ufw route allow in on eth1 out on eth2
```

This will allow all traffic routed to eth2 and coming in on eth1 to traverse the firewall.

```
ufw route allow in on eth0 out on eth1 to 12.34.45.67 port 80 proto tcp
```

This rule allows any packets coming in on eth0 to traverse the firewall out on eth1 to tcp port 80 on 12.34.45.67.

In addition to routing rules and policy, you must also setup IP forwarding. This may be done by setting the following in /etc/ufw/sysctl.conf:

```
net/ipv4/ip_forward=1
net/ipv6/conf/default/forwarding=1
net/ipv6/conf/all/forwarding=1
```

then restarting the firewall:

```
ufw disable
ufw enable
```

Be aware that setting kernel tunables is operating system specific and **ufw** sysctl settings may be overridden. See the **sysctl** manual page for details.

ufw supports connection rate limiting, which is useful for protecting against brute-force login attacks. When a limit rule is used, **ufw** will normally allow the connection but will deny connections if an IP address attempts to initiate 6 or more connections within 30 seconds. See <http://www.debian-administration.org/articles/187> for details. Typical usage is:

```
ufw limit ssh/tcp
```

Sometimes it is desirable to let the sender know when traffic is being denied, rather than simply ignoring it. In these cases, use **reject** instead of **deny**. For example:

```
ufw reject auth
```

By default, **ufw** will apply rules to all available interfaces. To limit this, specify **DIRECTION on INTERFACE**, where DIRECTION is one of **in** or **out** (interface aliases are not supported). For example, to allow all new incoming http connections on eth0, use:

```
ufw allow in on eth0 to any port 80 proto tcp
```

To delete a rule, simply prefix the original rule with **delete** with or without the rule comment. For example, if the original rule was:

```
ufw deny 80/tcp
```

Use this to delete it:

```
ufw delete deny 80/tcp
```

You may also specify the rule by NUM, as seen in the **status numbered** output. For example, if you want to delete rule number '3', use:

```
ufw delete 3
```

If you have IPv6 enabled and are deleting a generic rule that applies to both IPv4 and IPv6 (eg 'ufw allow 22/tcp'), deleting by rule number will delete only the specified rule. To delete both with one command, prefix the original rule with **delete**.

To insert a rule, specify the new rule as normal, but prefix the rule with the rule number to insert. For example, if you have four rules, and you want to insert a new rule as rule number three, use:

```
ufw insert 3 deny to any port 22 from 10.0.0.135 proto tcp
```

Similarly, to add a rule before all other rules matching the rule's IP type, use the **prepend** rule:

```
ufw prepend deny from 1.2.3.4
```

This is particularly useful for dynamic firewalls as found in an IPS. Importantly, if the specified rule is an IPv4 rule, it will be prepended before all other IPv4 rules. If it is an IPv6 rule, it will be prepended before any IPv6 rules.

To see a list of numbered rules, use:

```
ufw status numbered
```

ufw supports per rule logging. By default, no logging is performed when a packet matches a rule. Specifying **log** will log all new connections matching the rule, and **log-all** will log all packets matching the rule. For example, to allow and log all new ssh connections, use:

```
ufw allow log 22/tcp
```

See **LOGGING** for more information on logging.

EXAMPLES

Deny all access to port 53:

```
ufw deny 53
```

Allow all access to tcp port 80:

```
ufw allow 80/tcp
```

Allow all access from RFC1918 networks to this host:

```
ufw allow from 10.0.0.0/8  
ufw allow from 172.16.0.0/12
```

```
ufw allow from 192.168.0.0/16
```

Deny access to udp port 514 from host 1.2.3.4:

```
ufw deny proto udp from 1.2.3.4 to any port 514
```

Allow access to udp 1.2.3.4 port 5469 from 1.2.3.5 port 5469:

```
ufw allow proto udp from 1.2.3.5 port 5469 to 1.2.3.4 port 5469
```

REMOTE MANAGEMENT

When running **ufw enable** or starting **ufw** via its initscript, **ufw** will flush its chains. This is required so **ufw** can maintain a consistent state, but it may drop existing connections (eg ssh). **ufw** does support adding rules before enabling the firewall, so administrators can do:

```
ufw allow proto tcp from any to any port 22
```

before running '**ufw enable**'. The rules will still be flushed, but the ssh port will be open after enabling the firewall. Please note that once ufw is 'enabled', **ufw** will not flush the chains when adding or removing rules (but will when modifying a rule or changing the default policy). By default, **ufw** will prompt when enabling the firewall while running under ssh. This can be disabled by using '**ufw --force enable**'.

APPLICATION INTEGRATION

ufw supports application integration by reading profiles located in /etc/ufw/applications.d. To list the names of application profiles known to **ufw**, use:

```
ufw app list
```

Users can specify an application name when adding a rule (quoting any profile names with spaces). For example, when using the simple syntax, users can use:

```
ufw allow <name>
```

Or for the extended syntax:

```
ufw allow from 192.168.0.0/16 to any app <name>
```

You should not specify the protocol with either syntax, and with the extended syntax, use **app** in place of the **port** clause.

Details on the firewall profile for a given application can be seen with:

```
ufw app info <name>
```

where '<name>' is one of the applications seen with the app list command. Users may also specify **all** to see the profiles for all known applications.

Syntax for the application profiles is a simple .INI format:

```
[<name>]
title=<title>
description=<description>
ports=<ports>
```

The 'ports' field may specify a '|'-separated list of ports/protocols where the protocol is optional. A comma-separated list or a range (specified with 'start:end') may also be used to specify multiple ports, in which case the protocol is required. For example:

```
[SomeService]
title=Some title
description=Some description
ports=12/udp|34|56,78:90/tcp
```

In the above example, 'SomeService' may be used in app rules and it specifies UDP port 12, TCP and UDP on port 34 and TCP ports 56 and 78-90 inclusive.

After creating or editing an application profile, users can run:

```
ufw app update <name>
```

This command will automatically update the firewall with updated profile information. If specify 'all' for name, then all the profiles will be updated. To update a profile and add a new rule to the firewall automatically, users can run:

```
ufw app update --add-new <name>
```

The behavior of the **update --add-new** command can be configured using:

```
ufw app default <policy>
```

The default application policy is **skip**, which means that the **update --add-new** command will do nothing. Users may also specify a policy of **allow** or **deny** so the **update --add-new** command may automatically update the firewall. **WARNING:** it may be a security risk to use a default **allow** policy for application profiles. Carefully consider the security ramifications before using a default **allow** policy.

LOGGING

ufw supports multiple logging levels. **ufw** defaults to a loglevel of 'low' when a loglevel is not specified. Users may specify a loglevel with:

```
ufw logging LEVEL
```

LEVEL may be 'off', 'low', 'medium', 'high' and 'full'. Log levels are defined as:

off disables ufw managed logging

low logs all blocked packets not matching the defined policy (with rate limiting), as well as packets matching logged rules

medium

log level low, plus all allowed packets not matching the defined policy, all INVALID packets, and all new connections. All logging is done with rate limiting.

high log level medium (without rate limiting), plus all packets with rate limiting

full log level high without rate limiting

Loglevels above medium generate a lot of logging output, and may quickly fill up your disk. Loglevel medium may generate a lot of logging output on a busy system.

Specifying 'on' simply enables logging at log level 'low' if logging is currently not enabled.

REPORTS

The following reports are supported. Each is based on the live system and with the exception of the **listening** report, is in raw iptables format:

```
raw
builtins
before-rules
user-rules
after-rules
logging-rules
listening
added
```

The **raw** report shows the complete firewall, while the others show a subset of what is in the **raw** report.

The **listening** report will display the ports on the live system in the listening state for tcp and the open state for udp, along with the address of the interface and the executable listening on the port. An '*' is used in place of the address of the interface when the executable is bound to all interfaces on that port. Following this information is a list of rules which may affect connections on this port. The rules are listed in the order they are evaluated by the kernel, and the first match wins. Please note that the default policy is not listed and tcp6 and udp6 are shown only if IPV6 is enabled.

The **added** report displays the list of rules as they were added on the command-line. This report does not show the status of the running firewall (use '**ufw status**' instead). Because rules are normalized by **ufw**, rules may look different than the originally added rule. Also, **ufw** does not record command ordering, so an equivalent ordering is used which lists IPv6-only rules after other rules.

NOTES

On installation, **ufw** is disabled with a default incoming policy of deny, a default forward policy of deny, and a default outgoing policy of allow, with stateful tracking for NEW connections for incoming and forwarded connections. In addition to the above, a default ruleset is put in place that does the following:

- DROP packets with RH0 headers
- DROP INVALID packets
- ACCEPT certain icmp packets (INPUT and FORWARD): destination-unreachable, source-quench, time-exceeded, parameter-problem, and echo-request for IPv4. destination-unreachable, packet-too-big, time-exceeded, parameter-problem, and echo-request for IPv6.
- ACCEPT icmpv6 packets for stateless autoconfiguration (INPUT)
- ACCEPT ping replies from IPv6 link-local (ffe8::/10) addresses (INPUT)
- ACCEPT DHCP client traffic (INPUT)
- DROP non-local traffic (INPUT)
- ACCEPT mDNS (zeroconf/bonjour/avahi 224.0.0.251 for IPv4 and ff02::fb for IPv6) for service discovery (INPUT)
- ACCEPT UPnP (239.255.255.250 for IPv4 and ff02::f for IPv6) for service discovery (INPUT)

Rule ordering is important and the first match wins. Therefore when adding rules, add the more specific rules first with more general rules later.

ufw is not intended to provide complete firewall functionality via its command interface, but instead provides an easy way to add or remove simple rules.

The status command shows basic information about the state of the firewall, as well as rules managed via the **ufw** command. It does not show rules from the rules files in /etc/ufw. To see the complete state of the

firewall, users can **ufw show raw**. This displays the filter, nat, mangle and raw tables using:

```
iptables -n -L -v -x -t <table>
ip6tables -n -L -v -x -t <table>
```

See the **iptables** and **ip6tables** documentation for more details.

If the default policy is set to REJECT, **ufw** may interfere with rules added outside of the ufw framework. See README for details.

IPv6 is allowed by default. To change this behavior to only accept IPv6 traffic on the loopback interface, set IPV6 to 'no' in /etc/default/ufw and reload **ufw**. When IPv6 is enabled, you may specify rules in the same way as for IPv4 rules, and they will be displayed with **ufw status**. Rules that match both IPv4 and IPv6 addresses apply to both IP versions. For example, when IPv6 is enabled, the following rule will allow access to port 22 for both IPv4 and IPv6 traffic:

```
ufw allow 22
```

IPv6 over IPv4 tunnels and 6to4 are supported by using the 'ipv6' protocol ('41'). This protocol can only be used with the full syntax. For example:

```
ufw allow to 10.0.0.1 proto ipv6
ufw allow to 10.0.0.1 from 10.4.0.0/16 proto ipv6
```

IPSec is supported by using the 'esp' ('50') and 'ah' ('51') protocols. These protocols can only be used with the full syntax. For example:

```
ufw allow to 10.0.0.1 proto esp
ufw allow to 10.0.0.1 from 10.4.0.0/16 proto esp
ufw allow to 10.0.0.1 proto ah
ufw allow to 10.0.0.1 from 10.4.0.0/16 proto ah
```

In addition to the command-line interface, **ufw** also provides a framework which allows administrators to modify default behavior as well as take full advantage of netfilter. See the **ufw-framework** manual page for more information.

SEE ALSO

ufw-framework(8), **iptables(8)**, **ip6tables(8)**, **iptables-restore(8)**, **ip6tables-restore(8)**, **sysctl(8)**, **sysctl.conf(5)**

AUTHOR

ufw is Copyright 2008-2021, Canonical Ltd.

NAME

umount, umount2 – unmount filesystem

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/mount.h>
int umount(const char *target);
int umount2(const char *target, int flags);
```

DESCRIPTION

umount() and **umount2()** remove the attachment of the (topmost) filesystem mounted on *target*.

Appropriate privilege (Linux: the **CAP_SYS_ADMIN** capability) is required to unmount filesystems.

Linux 2.1.116 added the **umount2()** system call, which, like **umount()**, unmounts a target, but allows additional *flags* controlling the behavior of the operation:

MNT_FORCE (since Linux 2.1.116)

Ask the filesystem to abort pending requests before attempting the unmount. This may allow the unmount to complete without waiting for an inaccessible server, but could cause data loss. If, after aborting requests, some processes still have active references to the filesystem, the unmount will still fail. As at Linux 4.12, **MNT_FORCE** is supported only on the following filesystems: 9p (since Linux 2.6.16), ceph (since Linux 2.6.34), cifs (since Linux 2.6.12), fuse (since Linux 2.6.16), lustre (since Linux 3.11), and NFS (since Linux 2.1.116).

MNT_DETACH (since Linux 2.4.11)

Perform a lazy unmount: make the mount unavailable for new accesses, immediately disconnect the filesystem and all filesystems mounted below it from each other and from the mount table, and actually perform the unmount when the mount ceases to be busy.

MNT_EXPIRE (since Linux 2.6.8)

Mark the mount as expired. If a mount is not currently in use, then an initial call to **umount2()** with this flag fails with the error **EAGAIN**, but marks the mount as expired. The mount remains expired as long as it isn't accessed by any process. A second **umount2()** call specifying **MNT_EXPIRE** unmounts an expired mount. This flag cannot be specified with either **MNT_FORCE** or **MNT_DETACH**.

UMOUNT_NOFOLLOW (since Linux 2.6.34)

Don't dereference *target* if it is a symbolic link. This flag allows security problems to be avoided in set-user-ID-*root* programs that allow unprivileged users to unmount filesystems.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

The error values given below result from filesystem type independent errors. Each filesystem type may have its own special errors and its own special behavior. See the Linux kernel source code for details.

EAGAIN

A call to **umount2()** specifying **MNT_EXPIRE** successfully marked an unbusy filesystem as expired.

EBUSY

target could not be unmounted because it is busy.

EFAULT

target points outside the user address space.

EINVAL

target is not a mount point.

EINVAL

target is locked; see **mount_namespaces(7)**.

EINVAL

umount2() was called with **MNT_EXPIRE** and either **MNT_DETACH** or **MNT_FORCE**.

EINVAL (since Linux 2.6.34)

umount2() was called with an invalid flag value in *fla gs*.

ENAMETOOLONG

A pathname was longer than **MAXPATHLEN**.

ENOENT

A pathname was empty or had a nonexistent component.

ENOMEM

The kernel could not allocate a free page to copy filenames or data into.

EPERM

The caller does not have the required privileges.

VERSIONS

MNT_DETACH and **MNT_EXPIRE** are available since glibc 2.11.

STANDARDS

These functions are Linux-specific and should not be used in programs intended to be portable.

NOTES**umount() and shared mounts**

Shared mounts cause any mount activity on a mount, including **umount()** operations, to be forwarded to every shared mount in the peer group and every slave mount of that peer group. This means that **umount()** of any peer in a set of shared mounts will cause all of its peers to be unmounted and all of their slaves to be unmounted as well.

This propagation of unmount activity can be particularly surprising on systems where every mount is shared by default. On such systems, recursively bind mounting the root directory of the filesystem onto a subdirectory and then later unmounting that subdirectory with **MNT_DETACH** will cause every mount in the mount namespace to be lazily unmounted.

To ensure **umount()** does not propagate in this fashion, the mount may be remounted using a **mount(2)** call with a *mount_flags* argument that includes both **MS_REC** and **MS_PRIVATE** prior to **umount()** being called.

Historical details

The original **umount()** function was called as *umount(device)* and would return **ENOTBLK** when called with something other than a block device. In Linux 0.98p4, a call to *umount(dir)* was added, in order to support anonymous devices. In Linux 2.3.99-pre7, the call to *umount(device)* was removed, leaving only *umount(dir)* (since now devices can be mounted in more than one place, so specifying the device does not suffice).

SEE ALSO

mount(2), **mount_namespaces(7)**, **path_resolution(7)**, **mount(8)**, **umount(8)**

NAME

umount – unmount filesystems

SYNOPSIS

umount -a [-dflnrv] [-t fstype] [-O option...]

umount [-dflnrv] {directory|device}

umount -h|-V

DESCRIPTION

The **umount** command detaches the mentioned filesystem(s) from the file hierarchy. A filesystem is specified by giving the directory where it has been mounted. Giving the special device on which the filesystem lives may also work, but is obsolete, mainly because it will fail in case this device was mounted on more than one directory.

Note that a filesystem cannot be unmounted when it is 'busy' – for example, when there are open files on it, or when some process has its working directory there, or when a swap file on it is in use. The offending process could even be **umount** itself – it opens libc, and libc in its turn may open for example locale files. A lazy unmount avoids this problem, but it may introduce other issues. See **--lazy** description below.

OPTIONS**-a, --all**

All of the filesystems described in */proc/self/mountinfo* (or in deprecated */etc/mtab*) are unmounted, except the proc, devfs, devpts, sysfs, rpc_pipefs and nfsd filesystems. This list of the filesystems may be replaced by **--types** **umount** option.

-A, --all-targets

Unmount all mountpoints in the current mount namespace for the specified filesystem. The filesystem can be specified by one of the mountpoints or the device name (or UUID, etc.). When this option is used together with **--recursive**, then all nested mounts within the filesystem are recursively unmounted. This option is only supported on systems where */etc/mtab* is a symlink to */proc/mounts*.

-c, --no-canonicalize

Do not canonicalize paths. The paths canonicalization is based on **stat(2)** and **readlink(2)** system calls. These system calls may hang in some cases (for example on NFS if server is not available). The option has to be used with canonical path to the mount point.

This option is silently ignored by **umount** for non-root users.

For more details about this option see the **mount(8)** man page. Note that **umount** does not pass this option to the **/sbin/umount.type** helpers.

-d, --detach-loop

When the unmounted device was a loop device, also free this loop device. This option is unnecessary for devices initialized by **mount(8)**, in this case "autoclear" functionality is enabled by default.

--fake

Causes everything to be done except for the actual system call or umount helper execution; this 'fakes' unmounting the filesystem. It can be used to remove entries from the deprecated */etc/mtab* that were unmounted earlier with the **-n** option.

-f, --force

Force an unmount (in case of an unreachable NFS system).

Note that this option does not guarantee that umount command does not hang. It's strongly

recommended to use absolute paths without symlinks to avoid unwanted **readlink(2)** and **stat(2)** system calls on unreachable NFS in **umount**.

-i, --internal-only

Do not call the **/sbin/umount**,*filesystem* helper even if it exists. By default such a helper program is called if it exists.

-l, --lazy

Lazy unmount. Detach the filesystem from the file hierarchy now, and clean up all references to this filesystem as soon as it is not busy anymore.

A system reboot would be expected in near future if you're going to use this option for network filesystem or local filesystem with submounts. The recommended use-case for **umount -l** is to prevent hangs on shutdown due to an unreachable network share where a normal **umount** will hang due to a downed server or a network partition. Remounts of the share will not be possible.

-N, --namespace *ns*

Perform **umount** in the mount namespace specified by *ns*. *ns* is either PID of process running in that namespace or special file representing that namespace.

umount switches to the namespace when it reads */etc/fstab*, writes */etc/mtab* (or writes to */run/mount*) and calls **umount(2)** system call, otherwise it runs in the original namespace. It means that the target mount namespace does not have to contain any libraries or other requirements necessary to execute **umount(2)** command.

See **mount_namespaces(7)** for more information.

-n, --no-mtab

Unmount without writing in */etc/mtab*.

-O, --test-opts *option...*

Unmount only the filesystems that have the specified option set in */etc/fstab*. More than one option may be specified in a comma-separated list. Each option can be prefixed with **no** to indicate that no action should be taken for this option.

-q, --quiet

Suppress "not mounted" error messages.

-R, --recursive

Recursively unmount each specified directory. Recursion for each directory will stop if any unmount operation in the chain fails for any reason. The relationship between mountpoints is determined by */proc/self/mountinfo* entries. The filesystem must be specified by mountpoint path; a recursive unmount by device name (or UUID) is unsupported. Since version 2.37 it umounts also all over-mounted filesystems (more filesystems on the same mountpoint).

-r, --read-only

When an unmount fails, try to remount the filesystem read-only.

-t, --types *type...*

Indicate that the actions should only be taken on filesystems of the specified *type*. More than one type may be specified in a comma-separated list. The list of filesystem types can be prefixed with **no** to indicate that no action should be taken for all of the mentioned types. Note that **umount** reads information about mounted filesystems from kernel (*/proc/mounts*) and filesystem names may be different than filesystem names used in the */etc/fstab* (e.g., "nfs4" vs. "nfs").

-v, --verbose
Verbose mode.

-h, --help
Display help text and exit.

-V, --version
Print version and exit.

NON-SUPERUSER UMTOMTS

Normally, only the superuser can umount filesystems. However, when *fstab* contains the **user** option on a line, anybody can umount the corresponding filesystem. For more details see **mount(8)** man page.

Since version 2.34 the **umount** command can be used to perform umount operation also for fuse filesystems if kernel mount table contains user's ID. In this case *fstab* **user=** mount option is not required.

Since version 2.35 **umount** command does not exit when user permissions are inadequate by internal **libmount** security rules. It drops suid permissions and continue as regular non-root user. This can be used to support use-cases where root permissions are not necessary (e.g., fuse filesystems, user namespaces, etc).

LOOP DEVICE

The **umount** command will automatically detach loop device previously initialized by **mount(8)** command independently of */etc/mtab*.

In this case the device is initialized with "autoclear" flag (see **losetup(8)** output for more details), otherwise it's necessary to use the option **--detach-loop** or call **losetup -d device**. The autoclear feature is supported since Linux 2.6.25.

EXTERNAL HELPERS

The syntax of external umount helpers is:

```
umount.suffix {directory|device} [-flnrv] [-N namespace] [-t type.subtype]
```

where *suffix* is the filesystem type (or the value from a **uhelper=** or **helper=** marker in the *mtab* file). The **-t** option can be used for filesystems that have subtype support. For example:

```
umount.fuse -t fuse.sshfs
```

A **uhelper=***something* marker (unprivileged helper) can appear in the */etc/mtab* file when ordinary users need to be able to umount a mountpoint that is not defined in */etc/fstab* (for example for a device that was mounted by **udisks(1)**).

A **helper=***type* marker in the *mtab* file will redirect all umount requests to the **/sbin/umount.type** helper independently of UID.

Note that */etc/mtab* is currently deprecated and **helper=** and other userspace mount options are maintained by **libmount**.

ENVIRONMENT

LIBMOUNT_FSTAB=<path>
overrides the default location of the *fstab* file (ignored for **suid**)

LIBMOUNT_MTAB=<path>
overrides the default location of the *mtab* file (ignored for **suid**)

LIBMOUNT_DEBUG=all

enables **libmount** debug output

FILES

/etc/mtab

table of mounted filesystems (deprecated and usually replaced by symlink to */proc/mounts*)

/etc/fstab

table of known filesystems

/proc/self/mountinfo

table of mounted filesystems generated by kernel.

HISTORY

A **umount** command appeared in Version 6 AT&T UNIX.

SEE ALSO

umount(2), losetup(8), mount_namespaces(7), mount(8)

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **umount** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

umount.nfs(8) - Linux man page

Name

umount.nfs, umount.nfs4 - unmount a Network File System

Synopsis

umount.nfs *dir* [**-fvnrlh**]

Description

umount.nfs and **umount.nfs4** are a part of [nfs\(5\)](#) utilities package, which provides NFS client functionality.

umount.nfs4 and **umount.nfs** are meant to be used by the [umount\(8\)](#) command for unmounting NFS shares. This subcommand, however, can also be used as a standalone command with limited functionality.

dir is the directory on which the file system is mounted.

Options

-f

Force unmount the file system in case of unreachable NFS system.

-v

Be verbose.

-n

Do not update */etc/mtab*. By default, an entry is created in */etc/mtab* for every mounted file system. Use this option to skip deleting an entry.

-r

In case unmounting fails, try to mount read-only.

-l

Lazy unmount. Detach the file system from the file system hierarchy now, and cleanup all references to the file system as soon as it is not busy anymore.

-h

Print help message.

Note

For further information please refer [**nfs**\(5\)](#) and [**umount**\(8\)](#) manual pages.

Files

/etc/fstab

file system table

/etc/mtab

table of mounted file systems

See Also

[**nfs**\(5\)](#), [**umount**\(8\)](#),

Author

Amit Gud <agud@redhat.com>

NAME

umount.udisks2 – unmount file systems that have been mounted by UDisks2

DESCRIPTION

The **umount.udisks2** program is a helper for the **umount(8)** program. Its purpose is to clean up automatically created directories created at file system mount-time. It should never be called directly.

AUTHOR

This man page was originally written for UDisks2 by David Zeuthen <zeuthen@gmail.com> with a lot of help from many others.

BUGS

Please send bug reports to either the distribution bug tracker or the upstream bug tracker at
<https://github.com/storaged-project/udisks/issues>.

SEE ALSO

udisks(8), udisksd(8), udisksctl(1), umount(8),

NAME

umount, umount2 – unmount filesystem

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/mount.h>
int umount(const char *target);
int umount2(const char *target, int flags);
```

DESCRIPTION

umount() and **umount2()** remove the attachment of the (topmost) filesystem mounted on *target*.

Appropriate privilege (Linux: the **CAP_SYS_ADMIN** capability) is required to unmount filesystems.

Linux 2.1.116 added the **umount2()** system call, which, like **umount()**, unmounts a target, but allows additional *flags* controlling the behavior of the operation:

MNT_FORCE (since Linux 2.1.116)

Ask the filesystem to abort pending requests before attempting the unmount. This may allow the unmount to complete without waiting for an inaccessible server, but could cause data loss. If, after aborting requests, some processes still have active references to the filesystem, the unmount will still fail. As at Linux 4.12, **MNT_FORCE** is supported only on the following filesystems: 9p (since Linux 2.6.16), ceph (since Linux 2.6.34), cifs (since Linux 2.6.12), fuse (since Linux 2.6.16), lustre (since Linux 3.11), and NFS (since Linux 2.1.116).

MNT_DETACH (since Linux 2.4.11)

Perform a lazy unmount: make the mount unavailable for new accesses, immediately disconnect the filesystem and all filesystems mounted below it from each other and from the mount table, and actually perform the unmount when the mount ceases to be busy.

MNT_EXPIRE (since Linux 2.6.8)

Mark the mount as expired. If a mount is not currently in use, then an initial call to **umount2()** with this flag fails with the error **EAGAIN**, but marks the mount as expired. The mount remains expired as long as it isn't accessed by any process. A second **umount2()** call specifying **MNT_EXPIRE** unmounts an expired mount. This flag cannot be specified with either **MNT_FORCE** or **MNT_DETACH**.

UMOUNT_NOFOLLOW (since Linux 2.6.34)

Don't dereference *target* if it is a symbolic link. This flag allows security problems to be avoided in set-user-ID-*root* programs that allow unprivileged users to unmount filesystems.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS

The error values given below result from filesystem type independent errors. Each filesystem type may have its own special errors and its own special behavior. See the Linux kernel source code for details.

EAGAIN

A call to **umount2()** specifying **MNT_EXPIRE** successfully marked an unbusy filesystem as expired.

EBUSY

target could not be unmounted because it is busy.

EFAULT

target points outside the user address space.

EINVAL

target is not a mount point.

EINVAL

target is locked; see **mount_namespaces(7)**.

EINVAL

umount2() was called with **MNT_EXPIRE** and either **MNT_DETACH** or **MNT_FORCE**.

EINVAL (since Linux 2.6.34)

umount2() was called with an invalid flag value in *fla gs*.

ENAMETOOLONG

A pathname was longer than **MAXPATHLEN**.

ENOENT

A pathname was empty or had a nonexistent component.

ENOMEM

The kernel could not allocate a free page to copy filenames or data into.

EPERM

The caller does not have the required privileges.

VERSIONS

MNT_DETACH and **MNT_EXPIRE** are available since glibc 2.11.

STANDARDS

These functions are Linux-specific and should not be used in programs intended to be portable.

NOTES**umount() and shared mounts**

Shared mounts cause any mount activity on a mount, including **umount()** operations, to be forwarded to every shared mount in the peer group and every slave mount of that peer group. This means that **umount()** of any peer in a set of shared mounts will cause all of its peers to be unmounted and all of their slaves to be unmounted as well.

This propagation of unmount activity can be particularly surprising on systems where every mount is shared by default. On such systems, recursively bind mounting the root directory of the filesystem onto a subdirectory and then later unmounting that subdirectory with **MNT_DETACH** will cause every mount in the mount namespace to be lazily unmounted.

To ensure **umount()** does not propagate in this fashion, the mount may be remounted using a **mount(2)** call with a *mount_flags* argument that includes both **MS_REC** and **MS_PRIVATE** prior to **umount()** being called.

Historical details

The original **umount()** function was called as *umount(device)* and would return **ENOTBLK** when called with something other than a block device. In Linux 0.98p4, a call to *umount(dir)* was added, in order to support anonymous devices. In Linux 2.3.99-pre7, the call to *umount(device)* was removed, leaving only *umount(dir)* (since now devices can be mounted in more than one place, so specifying the device does not suffice).

SEE ALSO

mount(2), **mount_namespaces(7)**, **path_resolution(7)**, **mount(8)**, **umount(8)**

NAME

uname – print system information

SYNOPSIS

uname [*OPTION*]...

DESCRIPTION

Print certain system information. With no OPTION, same as **-s**.

-a, --all

print all information, in the following order, except omit **-p** and **-i** if unknown:

-s, --kernel-name

print the kernel name

-n, --nodename

print the network node hostname

-r, --kernel-release

print the kernel release

-v, --kernel-version

print the kernel version

-m, --machine

print the machine hardware name

-p, --processor

print the processor type (non–portable)

-i, --hardware-platform

print the hardware platform (non–portable)

-o, --operating-system

print the operating system

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

arch(1), uname(2)

Full documentation <<https://www.gnu.org/software/coreutils/uname>>
or available locally via: info '(coreutils) uname invocation'

NAME

`uname` – get name and information about current kernel

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <sys/utsname.h>
int uname(struct utsname *buf);
```

DESCRIPTION

`uname()` returns system information in the structure pointed to by *buf*. The *utsname* struct is defined in `<sys/utsname.h>`:

```
struct utsname {
    char sysname[];      /* Operating system name (e.g., "Linux") */
    char nodename[];     /* Name within communications network
                           to which the node is attached, if any */
    char release[];      /* Operating system release
                           (e.g., "2.6.28") */
    char version[];      /* Operating system version */
    char machine[];      /* Hardware type identifier */

#ifndef _GNU_SOURCE
    char domainname[];  /* NIS or YP domain name */
#endif
};
```

The length of the arrays in a *struct utsname* is unspecified (see NOTES); the fields are terminated by a null byte ('\0').

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and *errno* is set to indicate the error.

ERRORS**FAULT**

buf is not valid.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.4BSD.

The *domainname* member (the NIS or YP domain name) is a GNU extension.

NOTES

The kernel has the name, release, version, and supported machine type built in. Conversely, the *nodename* field is configured by the administrator to match the network (this is what the BSD historically calls the "hostname", and is set via `sethostname(2)`). Similarly, the *domainname* field is set via `setdomainname(2)`.

The length of the fields in the struct varies. Some operating systems or libraries use a hardcoded 9 or 33 or 65 or 257. Other systems use **SYS_NMLN** or **_SYS_NMLN** or **UTSLEN** or **_UTSNAME_LENGTH**. Clearly, it is a bad idea to use any of these constants; just use `sizeof(...)`. SVr4 uses 257, "to support Internet hostnames" — this is the largest value likely to be encountered in the wild.

Part of the *utsname* information is also accessible via `/proc/sys/kernel/{ostype, hostname, osrelease, version, domainname}`.

C library/kernel differences

Over time, increases in the size of the *utsname* structure have led to three successive versions of `uname()`: `sys_olduname()` (slot `__NR_oldolduname`), `sys_uname()` (slot `__NR_olduname`), and `sys_newuname()` (slot `__NR_uname`). The first one used length 9 for all fields; the second used 65; the third also uses 65 but adds the *domainname* field. The glibc `uname()` wrapper function hides these details from applications, invoking the most recent version of the system call provided by the kernel.

SEE ALSO

uname(1), getdomainname(2), gethostname(2), uts_namespaces(7)

NAME

uniq – report or omit repeated lines

SYNOPSIS

uniq [*OPTION*]... [*INPUT* [*OUTPUT*]]

DESCRIPTION

Filter adjacent matching lines from *INPUT* (or standard input), writing to *OUTPUT* (or standard output).

With no options, matching lines are merged to the first occurrence.

Mandatory arguments to long options are mandatory for short options too.

-c, --count

prefix lines by the number of occurrences

-d, --repeated

only print duplicate lines, one for each group

-D

print all duplicate lines

--all-repeated[=METHOD]

like **-D**, but allow separating groups with an empty line; METHOD={none(default),prepend,separate}

-f, --skip-fields=N

avoid comparing the first N fields

--group[=METHOD]

show all items, separating groups with an empty line; METHOD={separate(default),prepend,append,both}

-i, --ignore-case

ignore differences in case when comparing

-s, --skip-chars=N

avoid comparing the first N characters

-u, --unique

only print unique lines

-z, --zero-terminated

line delimiter is NUL, not newline

-w, --check-chars=N

compare no more than N characters in lines

--help display this help and exit**--version**

output version information and exit

A field is a run of blanks (usually spaces and/or TABs), then non-blank characters. Fields are skipped before chars.

Note: 'uniq' does not detect repeated lines unless they are adjacent. You may want to sort the input first, or use 'sort -u' without 'uniq'.

AUTHOR

Written by Richard M. Stallman and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

comm(1), join(1), sort(1)

Full documentation <<https://www.gnu.org/software/coreutils/uniq>>
or available locally via: info '(coreutils) uniq invocation'

NAME

unzip – list, test and extract compressed files in a ZIP archive

SYNOPSIS

unzip [-Z] [-cflptTuvz[abjnoqsCDKLMUVWX\$/:^]] file[.zip] [file(s) ...] [-x xfile(s) ...] [-d exdir]

DESCRIPTION

unzip will list, test, or extract files from a ZIP archive, commonly found on MS-DOS systems. The default behavior (with no options) is to extract into the current directory (and subdirectories below it) all files from the specified ZIP archive. A companion program, *zip(1)*, creates ZIP archives; both programs are compatible with archives created by PKWARE's *PKZIP* and *PKUNZIP* for MS-DOS, but in many cases the program options or default behaviors differ.

ARGUMENTS

file[.zip]

Path of the ZIP archive(s). If the file specification is a wildcard, each matching file is processed in an order determined by the operating system (or file system). Only the filename can be a wildcard; the path itself cannot. Wildcard expressions are similar to those supported in commonly used Unix shells (*sh*, *ksh*, *csh*) and may contain:

- * matches a sequence of 0 or more characters
- ? matches exactly 1 character
- [...] matches any single character found inside the brackets; ranges are specified by a beginning character, a hyphen, and an ending character. If an exclamation point or a caret ('!' or '^') follows the left bracket, then the range of characters within the brackets is complemented (that is, anything *except* the characters inside the brackets is considered a match). To specify a verbatim left bracket, the three-character sequence "[[]]" has to be used.

(Be sure to quote any character that might otherwise be interpreted or modified by the operating system, particularly under Unix and VMS.) If no matches are found, the specification is assumed to be a literal filename; and if that also fails, the suffix *.zip* is appended. Note that self-extracting ZIP files are supported, as with any other ZIP archive; just specify the *.exe* suffix (if any) explicitly.

[file(s)] An optional list of archive members to be processed, separated by spaces. (VMS versions compiled with VMSCLI defined must delimit files with commas instead. See **-v** in **OPTIONS** below.) Regular expressions (wildcards) may be used to match multiple members; see above. Again, be sure to quote expressions that would otherwise be expanded or modified by the operating system.

[-x xfile(s)]

An optional list of archive members to be excluded from processing. Since wildcard characters normally match ('/') directory separators (for exceptions see the option **-W**), this option may be used to exclude any files that are in subdirectories. For example, "unzip foo *.[ch] -x */*" would extract all C source files in the main directory, but none in any subdirectories. Without the **-x** option, all C source files in all directories within the zipfile would be extracted.

[-d exdir]

An optional directory to which to extract files. By default, all files and subdirectories are recreated in the current directory; the **-d** option allows extraction in an arbitrary directory (always assuming one has permission to write to the directory). This option need not appear at the end of the command line; it is also accepted before the zipfile specification (with the normal options), immediately after the zipfile specification, or between the *file(s)* and the **-x** option. The option and directory may be concatenated without any white space between them, but note that this may cause normal shell behavior to be suppressed. In particular, "-d ~" (tilde) is expanded by Unix C shells into the name of the user's home directory, but "-d ~" is treated as a literal subdirectory "~" of the current directory.

OPTIONS

Note that, in order to support obsolescent hardware, *unzip*'s usage screen is limited to 22 or 23 lines and should therefore be considered only a reminder of the basic *unzip* syntax rather than an exhaustive list of all possible flags. The exhaustive list follows:

- Z** *zipinfo(1)* mode. If the first option on the command line is **-Z**, the remaining options are taken to be *zipinfo(1)* options. See the appropriate manual page for a description of these options.
- A** [OS/2, Unix DLL] print extended help for the DLL's programming interface (API).
- c** extract files to stdout/screen ("CRT"). This option is similar to the **-p** option except that the name of each file is printed as it is extracted, the **-a** option is allowed, and ASCII-EBCDIC conversion is automatically performed if appropriate. This option is not listed in the *unzip* usage screen.
- f** freshen existing files, i.e., extract only those files that already exist on disk and that are newer than the disk copies. By default *unzip* queries before overwriting, but the **-o** option may be used to suppress the queries. Note that under many operating systems, the TZ (timezone) environment variable must be set correctly in order for **-f** and **-u** to work properly (under Unix the variable is usually set automatically). The reasons for this are somewhat subtle but have to do with the differences between DOS-format file times (always local time) and Unix-format times (always in GMT/UTC) and the necessity to compare the two. A typical TZ value is "PST8PDT" (US Pacific time with automatic adjustment for Daylight Savings Time or "summer time").
- l** list archive files (short format). The names, uncompressed file sizes and modification dates and times of the specified files are printed, along with totals for all files specified. If UnZip was compiled with OS2_EAS defined, the **-l** option also lists columns for the sizes of stored OS/2 extended attributes (EAs) and OS/2 access control lists (ACLs). In addition, the zipfile comment and individual file comments (if any) are displayed. If a file was archived from a single-case file system (for example, the old MS-DOS FAT file system) and the **-L** option was given, the filename is converted to lowercase and is prefixed with a caret (^).
- p** extract files to pipe (stdout). Nothing but the file data is sent to stdout, and the files are always extracted in binary format, just as they are stored (no conversions).
- t** test archive files. This option extracts each specified file in memory and compares the CRC (cyclic redundancy check, an enhanced checksum) of the expanded file with the original file's stored CRC value.
- T** [most OSes] set the timestamp on the archive(s) to that of the newest file in each one. This corresponds to *zip*'s **-go** option except that it can be used on wildcard zipfiles (e.g., "unzip -T *.zip") and is much faster.
- u** update existing files and create new ones if needed. This option performs the same function as the **-f** option, extracting (with query) files that are newer than those with the same name on disk, and in addition it extracts those files that do not already exist on disk. See **-f** above for information on setting the timezone properly.
- v** list archive files (verbose format) or show diagnostic version info. This option has evolved and now behaves as both an option and a modifier. As an option it has two purposes: when a zipfile is specified with no other options, **-v** lists archive files verbosely, adding to the basic **-l** info the compression method, compressed size, compression ratio and 32-bit CRC. In contrast to most of the competing utilities, *unzip* removes the 12 additional header bytes of encrypted entries from the compressed size numbers. Therefore, compressed size and compression ratio figures are independent of the entry's encryption status and show the correct compression performance. (The complete size of the encrypted compressed data stream for zipfile entries is reported by the more verbose *zipinfo(1)* reports, see the separate manual.) When no zipfile is specified (that is, the complete command is simply "unzip -v"), a diagnostic screen is printed. In addition to the normal header with release date and version, *unzip* lists the home Info-ZIP ftp site and where to find a list of other ftp and non-ftp sites; the target operating system for which it was compiled, as well as

(possibly) the hardware on which it was compiled, the compiler and version used, and the compilation date; any special compilation options that might affect the program's operation (see also **DECRYPTION** below); and any options stored in environment variables that might do the same (see **ENVIRONMENT OPTIONS** below). As a modifier it works in conjunction with other options (e.g., **-t**) to produce more verbose or debugging output; this is not yet fully implemented but will be in future releases.

- z** display only the archive comment.

MODIFIERS

- a** convert text files. Ordinarily all files are extracted exactly as they are stored (as "binary" files). The **-a** option causes files identified by *zip* as text files (those with the 't' label in *zipinfo* listings, rather than 'b') to be automatically extracted as such, converting line endings, end-of-file characters and the character set itself as necessary. (For example, Unix files use line feeds (LFs) for end-of-line (EOL) and have no end-of-file (EOF) marker; Macintoshes use carriage returns (CRs) for EOLs; and most PC operating systems use CR+LF for EOLs and control-Z for EOF. In addition, IBM mainframes and the Michigan Terminal System use EBCDIC rather than the more common ASCII character set, and NT supports Unicode.) Note that *zip*'s identification of text files is by no means perfect; some "text" files may actually be binary and vice versa. *unzip* therefore prints "[text]" or "[binary]" as a visual check for each file it extracts when using the **-a** option. The **-aa** option forces all files to be extracted as text, regardless of the supposed file type. On VMS, see also **-S**.
 - b** [general] treat all files as binary (no text conversions). This is a shortcut for---**a**.
 - b** [Tandem] force the creation files with filecode type 180 ('C') when extracting Zip entries marked as "text". (On Tandem, **-a** is enabled by default, see above).
 - b** [VMS] auto-convert binary files (see **-a** above) to fixed-length, 512-byte record format. Doubling the option (**-bb**) forces all files to be extracted in this format. When extracting to standard output (**-c** or **-p** option in effect), the default conversion of text record delimiters is disabled for binary (**-b**) resp. all (**-bb**) files.
 - B** [when compiled with UNIXBACKUP defined] save a backup copy of each overwritten file. The backup file is gets the name of the target file with a tilde and optionally a unique sequence number (up to 5 digits) appended. The sequence number is applied whenever another file with the original name plus tilde already exists. When used together with the "overwrite all" option **-o**, numbered backup files are never created. In this case, all backup files are named as the original file with an appended tilde, existing backup files are deleted without notice. This feature works similarly to the default behavior of *emacs*(1) in many locations.
- Example: the old copy of "foo" is renamed to "foo~".
- Warning: Users should be aware that the **-B** option does not prevent loss of existing data under all circumstances. For example, when *unzip* is run in overwrite-all mode, an existing "foo~" file is deleted before *unzip* attempts to rename "foo" to "foo~". When this rename attempt fails (because of a file locks, insufficient privileges, or ...), the extraction of "foo~" gets cancelled, but the old backup file is already lost. A similar scenario takes place when the sequence number range for numbered backup files gets exhausted (99999, or 65535 for 16-bit systems). In this case, the backup file with the maximum sequence number is deleted and replaced by the new backup version without notice.
- C** use case-insensitive matching for the selection of archive entries from the command-line list of extract selection patterns. *unzip*'s philosophy is "you get what you ask for" (this is also responsible for the **-L/-U** change; see the relevant options below). Because some file systems are fully case-sensitive (notably those under the Unix operating system) and because both ZIP archives and *unzip* itself are portable across platforms, *unzip*'s default behavior is to match both wildcard and literal filenames case-sensitively. That is, specifying "makefile" on the command line will *only* match "makefile" in the archive, not "Makefile" or "MAKEFILE" (and similarly for wildcard specifications). Since this does not correspond to the behavior of many other operating/file

systems (for example, OS/2 HPFS, which preserves mixed case but is not sensitive to it), the **-C** option may be used to force all filename matches to be case-insensitive. In the example above, all three files would then match “`makefile`” (or “`make*`”, or similar). The **-C** option affects file specs in both the normal file list and the excluded-file list (xlist).

Please note that the **-C** option does neither affect the search for the zipfile(s) nor the matching of archive entries to existing files on the extraction path. On a case-sensitive file system, *unzip* will never try to overwrite a file “`FOO`” when extracting an entry “`foo`”!

- D** skip restoration of timestamps for extracted items. Normally, *unzip* tries to restore all meta-information for extracted items that are supplied in the Zip archive (and do not require privileges or impose a security risk). By specifying **-D**, *unzip* is told to suppress restoration of timestamps for directories explicitly created from Zip archive entries. This option only applies to ports that support setting timestamps for directories (currently ATheOS, BeOS, MacOS, OS/2, Unix, VMS, Win32, for other *unzip* ports, **-D** has no effect). The duplicated option **-DD** forces suppression of timestamp restoration for all extracted entries (files and directories). This option results in setting the timestamps for all extracted entries to the current time.
- On VMS, the default setting for this option is **-D** for consistency with the behaviour of BACKUP: file timestamps are restored, timestamps of extracted directories are left at the current time. To enable restoration of directory timestamps, the negated option **--D** should be specified. On VMS, the option **-D** disables timestamp restoration for all extracted Zip archive items. (Here, a single **-D** on the command line combines with the default **-D** to do what an explicit **-DD** does on other systems.)
- E** [MacOS only] display contents of MacOS extra field during restore operation.
- F** [Acorn only] suppress removal of NFS filetype extension from stored filenames.
- F** [non-Acorn systems supporting long filenames with embedded commas, and only if compiled with ACORN_FTYPE_NFS defined] translate filetype information from ACORN RISC OS extra field blocks into a NFS filetype extension and append it to the names of the extracted files. (When the stored filename appears to already have an appended NFS filetype extension, it is replaced by the info from the extra field.)
- i** [MacOS only] ignore filenames stored in MacOS extra fields. Instead, the most compatible filename stored in the generic part of the entry’s header is used.
- j** junk paths. The archive’s directory structure is not recreated; all files are deposited in the extraction directory (by default, the current one).
- J** [BeOS only] junk file attributes. The file’s BeOS file attributes are not restored, just the file’s data.
- J** [MacOS only] ignore MacOS extra fields. All Macintosh specific info is skipped. Data-fork and resource-fork are restored as separate files.
- K** [AtheOS, BeOS, Unix only] retain SUID/SGID/Tacky file attributes. Without this flag, these attribute bits are cleared for security reasons.
- L** convert to lowercase any filename originating on an uppercase-only operating system or file system. (This was *unzip*’s default behavior in releases prior to 5.11; the new default behavior is identical to the old behavior with the **-U** option, which is now obsolete and will be removed in a future release.) Depending on the archiver, files archived under single-case file systems (VMS, old MS-DOS FAT, etc.) may be stored as all-uppercase names; this can be ugly or inconvenient when extracting to a case-preserving file system such as OS/2 HPFS or a case-sensitive one such as under Unix. By default *unzip* lists and extracts such filenames exactly as they’re stored (excepting truncation, conversion of unsupported characters, etc.); this option causes the names of all files from certain systems to be converted to lowercase. The **-LL** option forces conversion of every filename to lowercase, regardless of the originating file system.
- M** pipe all output through an internal pager similar to the Unix *more*(1) command. At the end of a screenful of output, *unzip* pauses with a “**--More--**” prompt; the next screenful may be viewed

by pressing the Enter (Return) key or the space bar. *unzip* can be terminated by pressing the “q” key and, on some systems, the Enter/Return key. Unlike Unix *more*(1), there is no forward-searching or editing capability. Also, *unzip* doesn’t notice if long lines wrap at the edge of the screen, effectively resulting in the printing of two or more lines and the likelihood that some text will scroll off the top of the screen before being viewed. On some systems the number of available lines on the screen is not detected, in which case *unzip* assumes the height is 24 lines.

- n** never overwrite existing files. If a file already exists, skip the extraction of that file without prompting. By default *unzip* queries before extracting any file that already exists; the user may choose to overwrite only the current file, overwrite all files, skip extraction of the current file, skip extraction of all existing files, or rename the current file.
- N** [Amiga] extract file comments as Amiga filenotes. File comments are created with the -c option of *zip*(1), or with the -N option of the Amiga port of *zip*(1), which stores filenotes as comments.
- o** overwrite existing files without prompting. This is a dangerous option, so use it with care. (It is often used with **-f**, however, and is the only way to overwrite directory EAs under OS/2.)
- P password** use *password* to decrypt encrypted zipfile entries (if any). **THIS IS INSECURE!** Many multi-user operating systems provide ways for any user to see the current command line of any other user; even on stand-alone systems there is always the threat of over-the-shoulder peeking. Storing the plaintext password as part of a command line in an automated script is even worse. Whenever possible, use the non-echoing, interactive prompt to enter passwords. (And where security is truly important, use strong encryption such as Pretty Good Privacy instead of the relatively weak encryption provided by standard zipfile utilities.)
- q** perform operations quietly (**-qq** = even quieter). Ordinarily *unzip* prints the names of the files it’s extracting or testing, the extraction methods, any file or zipfile comments that may be stored in the archive, and possibly a summary when finished with each archive. The **-q[q]** options suppress the printing of some or all of these messages.
- s** [OS/2, NT, MS-DOS] convert spaces in filenames to underscores. Since all PC operating systems allow spaces in filenames, *unzip* by default extracts filenames with spaces intact (e.g., “EA DATA. SF”). This can be awkward, however, since MS-DOS in particular does not gracefully support spaces in filenames. Conversion of spaces to underscores can eliminate the awkwardness in some cases.
- S** [VMS] convert text files (**-a**, **-aa**) into Stream_LF record format, instead of the text-file default, variable-length record format. (Stream_LF is the default record format of VMS *unzip*. It is applied unless conversion (**-a**, **-aa** and/or **-b**, **-bb**) is requested or a VMS-specific entry is processed.)
- U** [UNICODE_SUPPORT only] modify or disable UTF-8 handling. When UNICODE_SUPPORT is available, the option **-U** forces *unzip* to escape all non-ASCII characters from UTF-8 coded filenames as “#Uxxxx” (for UCS-2 characters, or “#Lxxxxxx” for unicode codepoints needing 3 octets). This option is mainly provided for debugging purpose when the fairly new UTF-8 support is suspected to mangle up extracted filenames.
The option **-UU** allows to entirely disable the recognition of UTF-8 encoded filenames. The handling of filename codings within *unzip* falls back to the behaviour of previous versions.
[old, obsolete usage] leave filenames uppercase if created under MS-DOS, VMS, etc. See **-L** above.
- V** retain (VMS) file version numbers. VMS files can be stored with a version number, in the format *file.ext ;##*. By default the “;##” version numbers are stripped, but this option allows them to be retained. (On file systems that limit filenames to particularly short lengths, the version numbers may be truncated or stripped regardless of this option.)
- W** [only when WILD_STOP_AT_DIR compile-time option enabled] modifies the pattern matching routine so that both ‘?’ (single-char wildcard) and ‘*’ (multi-char wildcard) do not match the directory separator character ‘/’. (The two-character sequence “**” acts as a multi-char wildcard

that includes the directory separator in its matched characters.) Examples:

```
"*.c" matches "foo.c" but not "mydir/foo.c"
"**.c" matches both "foo.c" and "mydir/foo.c"
"*/*.c" matches "bar/foo.c" but not "baz/bar/foo.c"
"?*/*" matches "ab/foo" and "abc/foo"
but not "a/foo" or "a/b/foo"
```

This modified behaviour is equivalent to the pattern matching style used by the shells of some of UnZip's supported target OSs (one example is Acorn RISC OS). This option may not be available on systems where the Zip archive's internal directory separator character '/' is allowed as regular character in native operating system filenames. (Currently, UnZip uses the same pattern matching rules for both wildcard zipfile specifications and zip entry selection patterns in most ports. For systems allowing '/' as regular filename character, the -W option would not work as expected on a wildcard zipfile specification.)

- X** [VMS, Unix, OS/2, NT, Tandem] restore owner/protection info (UICs and ACL entries) under VMS, or user and group info (UID/GID) under Unix, or access control lists (ACLs) under certain network-enabled versions of OS/2 (Warp Server with IBM LAN Server/Requester 3.0 to 5.0; Warp Connect with IBM Peer 1.0), or security ACLs under Windows NT. In most cases this will require special system privileges, and doubling the option (-XX) under NT instructs *unzip* to use privileges for extraction; but under Unix, for example, a user who belongs to several groups can restore files owned by any of those groups, as long as the user IDs match his or her own. Note that ordinary file attributes are always restored--this option applies only to optional, extra ownership info available on some operating systems. [NT's access control lists do not appear to be especially compatible with OS/2's, so no attempt is made at cross-platform portability of access privileges. It is not clear under what conditions this would ever be useful anyway.]
- Y** [VMS] treat archived file name endings of ".nnn" (where "nnn" is a decimal number) as if they were VMS version numbers (";nnn"). (The default is to treat them as file types.) Example:
"a.b.3" -> "a.b;3".
- \$** [MS-DOS, OS/2, NT] restore the volume label if the extraction medium is removable (e.g., a diskette). Doubling the option (-\$\$) allows fixed media (hard disks) to be labelled as well. By default, volume labels are ignored.
- / extensions** [Acorn only] overrides the extension list supplied by *Unzip\$Ext* environment variable. During extraction, filename extensions that match one of the items in this extension list are swapped in front of the base name of the extracted file.
- :** [all but Acorn, VM/CMS, MVS, Tandem] allows to extract archive members into locations outside of the current "extraction root folder". For security reasons, *unzip* normally removes "parent dir" path components ("..") from the names of extracted file. This safety feature (new for version 5.50) prevents *unzip* from accidentally writing files to "sensitive" areas outside the active extraction folder tree head. The -: option lets *unzip* switch back to its previous, more liberal behaviour, to allow exact extraction of (older) archives that used ".." components to create multiple directory trees at the level of the current extraction folder. This option does not enable writing explicitly to the root directory ("/"). To achieve this, it is necessary to set the extraction target folder to root (e.g. -d /). However, when the -: option is specified, it is still possible to implicitly write to the root directory by specifying enough ".." path components within the zip archive. Use this option with extreme caution.
- ^** [Unix only] allow control characters in names of extracted ZIP archive entries. On Unix, a file name may contain any (8-bit) character code with the two exception '/' (directory delimiter) and NUL (0x00, the C string termination indicator), unless the specific file system has more restrictive conventions. Generally, this allows to embed ASCII control characters (or even sophisticated control sequences) in file names, at least on 'native' Unix file systems. However, it may be highly suspicious to make use of this Unix "feature". Embedded control characters in file names might

have nasty side effects when displayed on screen by some listing code without sufficient filtering. And, for ordinary users, it may be difficult to handle such file names (e.g. when trying to specify it for open, copy, move, or delete operations). Therefore, *unzip* applies a filter by default that removes potentially dangerous control characters from the extracted file names. The **-^** option allows to override this filter in the rare case that embedded filename control characters are to be intentionally restored.

- 2** [VMS] force unconditionally conversion of file names to ODS2-compatible names. The default is to exploit the destination file system, preserving case and extended file name characters on an ODS5 destination file system; and applying the ODS2-compatibility file name filtering on an ODS2 destination file system.

ENVIRONMENT OPTIONS

unzip's default behavior may be modified via options placed in an environment variable. This can be done with any option, but it is probably most useful with the **-a**, **-L**, **-C**, **-q**, **-o**, or **-n** modifiers: make *unzip* auto-convert text files by default, make it convert filenames from uppercase systems to lowercase, make it match names case-insensitively, make it quieter, or make it always overwrite or never overwrite files as it extracts them. For example, to make *unzip* act as quietly as possible, only reporting errors, one would use one of the following commands:

Unix Bourne shell:

```
UNZIP=-qq; export UNZIP
```

Unix C shell:

```
setenv UNZIP -qq
```

OS/2 or MS-DOS:

```
set UNZIP=-qq
```

VMS (quotes for *lowercase*):

```
define UNZIP_OPTS "-qq"
```

Environment options are, in effect, considered to be just like any other command-line options, except that they are effectively the first options on the command line. To override an environment option, one may use the “minus operator” to remove it. For instance, to override one of the quiet-flags in the example above, use the command

```
unzip --q[other options] zipfile
```

The first hyphen is the normal switch character, and the second is a minus sign, acting on the q option. Thus the effect here is to cancel one quantum of quietness. To cancel both quiet flags, two (or more) minuses may be used:

```
unzip -t--q zipfile
unzip ---qt zipfile
```

(the two are equivalent). This may seem awkward or confusing, but it is reasonably intuitive: just ignore the first hyphen and go from there. It is also consistent with the behavior of Unix *nice*(1).

As suggested by the examples above, the default variable names are UNZIP_OPTS for VMS (where the symbol used to install *unzip* as a foreign command would otherwise be confused with the environment variable), and UNZIP for all other operating systems. For compatibility with *zip*(1), UNZIPOPT is also accepted (don't ask). If both UNZIP and UNZIPOPT are defined, however, UNZIP takes precedence. *unzip*'s diagnostic option (**-v** with no zipfile name) can be used to check the values of all four possible *unzip* and *zipinfo* environment variables.

The timezone variable (TZ) should be set according to the local timezone in order for the **-f** and **-u** to operate correctly. See the description of **-f** above for details. This variable may also be necessary to get timestamps of extracted files to be set correctly. The WIN32 (Win9x/ME/NT4/2K/XP/2K3) port of *unzip* gets the timezone configuration from the registry, assuming it is correctly set in the Control Panel. The TZ variable is ignored for this port.

DECRIPTION

Encrypted archives are fully supported by Info-ZIP software, but due to United States export restrictions, de-/encryption support might be disabled in your compiled binary. However, since spring 2000, US export restrictions have been liberated, and our source archives do now include full crypt code. In case you need binary distributions with crypt support enabled, see the file "WHERE" in any Info-ZIP source or binary distribution for locations both inside and outside the US.

Some compiled versions of *unzip* may not support decryption. To check a version for crypt support, either attempt to test or extract an encrypted archive, or else check *unzip*'s diagnostic screen (see the **-v** option above) for "[decryption]" as one of the special compilation options.

As noted above, the **-P** option may be used to supply a password on the command line, but at a cost in security. The preferred decryption method is simply to extract normally; if a zipfile member is encrypted, *unzip* will prompt for the password without echoing what is typed. *unzip* continues to use the same password as long as it appears to be valid, by testing a 12-byte header on each file. The correct password will always check out against the header, but there is a 1-in-256 chance that an incorrect password will as well. (This is a security feature of the PKWARE zipfile format; it helps prevent brute-force attacks that might otherwise gain a large speed advantage by testing only the header.) In the case that an incorrect password is given but it passes the header test anyway, either an incorrect CRC will be generated for the extracted data or else *unzip* will fail during the extraction because the "decrypted" bytes do not constitute a valid compressed data stream.

If the first password fails the header check on some file, *unzip* will prompt for another password, and so on until all files are extracted. If a password is not known, entering a null password (that is, just a carriage return or "Enter") is taken as a signal to skip all further prompting. Only unencrypted files in the archive(s) will thereafter be extracted. (In fact, that's not quite true; older versions of *zip(1)* and *zipcloak(1)* allowed null passwords, so *unzip* checks each encrypted file to see if the null password works. This may result in "false positives" and extraction errors, as noted above.)

Archives encrypted with 8-bit passwords (for example, passwords with accented European characters) may not be portable across systems and/or other archivers. This problem stems from the use of multiple encoding methods for such characters, including Latin-1 (ISO 8859-1) and OEM code page 850. DOS *PKZIP* 2.04g uses the OEM code page; Windows *PKZIP* 2.50 uses Latin-1 (and is therefore incompatible with DOS *PKZIP*); Info-ZIP uses the OEM code page on DOS, OS/2 and Win3.x ports but ISO coding (Latin-1 etc.) everywhere else; and Nico Mak's *WinZip* 6.x does not allow 8-bit passwords at all. *UnZip* 5.3 (or newer) attempts to use the default character set first (e.g., Latin-1), followed by the alternate one (e.g., OEM code page) to test passwords. On EBCDIC systems, if both of these fail, EBCDIC encoding will be tested as a last resort. (EBCDIC is not tested on non-EBCDIC systems, because there are no known archivers that encrypt using EBCDIC encoding.) ISO character encodings other than Latin-1 are not supported. The new addition of (partially) Unicode (resp. UTF-8) support in *UnZip* 6.0 has not yet been adapted to the encryption password handling in *unzip*. On systems that use UTF-8 as native character encoding, *unzip* simply tries decryption with the native UTF-8 encoded password; the built-in attempts to check the password in translated encoding have not yet been adapted for UTF-8 support and will consequently fail.

EXAMPLES

To use *unzip* to extract all members of the archive *letters.zip* into the current directory and subdirectories below it, creating any subdirectories as necessary:

```
unzip letters
```

To extract all members of *letters.zip* into the current directory only:

```
unzip -j letters
```

To test *letters.zip*, printing only a summary message indicating whether the archive is OK or not:

```
unzip -tq letters
```

To test *all* zipfiles in the current directory, printing only the summaries:

```
unzip -tq \*.zip
```

(The backslash before the asterisk is only required if the shell expands wildcards, as in Unix; double quotes could have been used instead, as in the source examples below.) To extract to standard output all members of *letters.zip* whose names end in *.tex*, auto-converting to the local end-of-line convention and piping the output into *more(1)*:

```
unzip -ca letters \*.tex | more
```

To extract the binary file *paper1.dvi* to standard output and pipe it to a printing program:

```
unzip -p articles paper1.dvi | dvips
```

To extract all FORTRAN and C source files--*.f, *.c, *.h, and Makefile--into the /tmp directory:

```
unzip source.zip ".*.[fch]" Makefile -d /tmp
```

(the double quotes are necessary only in Unix and only if globbing is turned on). To extract all FORTRAN and C source files, regardless of case (e.g., both *.c and *.C, and any makefile, Makefile, MAKEFILE or similar):

```
unzip -C source.zip ".*.[fch]" makefile -d /tmp
```

To extract any such files but convert any uppercase MS-DOS or VMS names to lowercase and convert the line-endings of all of the files to the local standard (without respect to any files that might be marked “binary”):

```
unzip -aaCL source.zip ".*.[fch]" makefile -d /tmp
```

To extract only newer versions of the files already in the current directory, without querying (NOTE: be careful of unzipping in one timezone a zipfile created in another--ZIP archives other than those created by Zip 2.1 or later contain no timezone information, and a “newer” file from an eastern timezone may, in fact, be older):

```
unzip -fo sources
```

To extract newer versions of the files already in the current directory and to create any files not already there (same caveat as previous example):

```
unzip -uo sources
```

To display a diagnostic screen showing which *unzip* and *zipinfo* options are stored in environment variables, whether decryption support was compiled in, the compiler with which *unzip* was compiled, etc.:

```
unzip -v
```

In the last five examples, assume that UNZIP or UNZIP_OPTS is set to -q. To do a singly quiet listing:

```
unzip -l file.zip
```

To do a doubly quiet listing:

```
unzip -ql file.zip
```

(Note that the “.zip” is generally not necessary.) To do a standard listing:

```
unzip --ql file.zip
```

or

```
unzip -l-q file.zip
```

or

```
unzip -l--q file.zip
```

(Extra minuses in options don’t hurt.)

TIPS

The current maintainer, being a lazy sort, finds it very useful to define a pair of aliases: *tt* for ‘‘*unzip -tq*’’ and *ii* for ‘‘*unzip -Z*’’ (or ‘‘*zipinfo*’’). One may then simply type ‘‘*tt zipfile*’’ to test an archive, something that is worth making a habit of doing. With luck *unzip* will report ‘‘No errors detected in compressed data of *zipfile.zip*,’’ after which one may breathe a sigh of relief.

The maintainer also finds it useful to set the UNZIP environment variable to “-aL” and is tempted to add

“-C” as well. His ZIPINFO variable is set to “-z”.

DIAGNOSTICS

The exit status (or error level) approximates the exit codes defined by PKWARE and takes on the following values, except under VMS:

- 0 normal; no errors or warnings detected.
- 1 one or more warning errors were encountered, but processing completed successfully anyway. This includes zipfiles where one or more files was skipped due to unsupported compression method or encryption with an unknown password.
- 2 a generic error in the zipfile format was detected. Processing may have completed successfully anyway; some broken zipfiles created by other archivers have simple workarounds.
- 3 a severe error in the zipfile format was detected. Processing probably failed immediately.
- 4 *unzip* was unable to allocate memory for one or more buffers during program initialization.
- 5 *unzip* was unable to allocate memory or unable to obtain a tty to read the decryption password(s).
- 6 *unzip* was unable to allocate memory during decompression to disk.
- 7 *unzip* was unable to allocate memory during in-memory decompression.
- 8 [currently not used]
- 9 the specified zipfiles were not found.
- 10 invalid options were specified on the command line.
- 11 no matching files were found.
- 50 the disk is (or was) full during extraction.
- 51 the end of the ZIP archive was encountered prematurely.
- 80 the user aborted *unzip* prematurely with control-C (or similar)
- 81 testing or extraction of one or more files failed due to unsupported compression methods or unsupported decryption.
- 82 no files were found due to bad decryption password(s). (If even one file is successfully processed, however, the exit status is 1.)

VMS interprets standard Unix (or PC) return values as other, scarier-looking things, so *unzip* instead maps them into VMS-style status codes. The current mapping is as follows: 1 (success) for normal exit, 0x7fff0001 for warning errors, and (0x7fff000? + 16*normal_unzip_exit_status) for all other errors, where the ‘?’ is 2 (error) for *unzip* values 2, 9-11 and 80-82, and 4 (fatal error) for the remaining ones (3-8, 50, 51). In addition, there is a compilation option to expand upon this behavior: defining RETURN_CODES results in a human-readable explanation of what the error status means.

BUGS

Multi-part archives are not yet supported, except in conjunction with *zip*. (All parts must be concatenated together in order, and then “*zip -F*” (for *zip 2.x*) or “*zip -FF*” (for *zip 3.x*) must be performed on the concatenated archive in order to “fix” it. Also, *zip 3.0* and later can combine multi-part (split) archives into a combined single-file archive using “*zip -s- inarchive -O outarchive*”. See the *zip 3* manual page for more information.) This will definitely be corrected in the next major release.

Archives read from standard input are not yet supported, except with *funzip* (and then only the first member of the archive can be extracted).

Archives encrypted with 8-bit passwords (e.g., passwords with accented European characters) may not be portable across systems and/or other archivers. See the discussion in **DECRYPTION** above.

unzip's **-M** ("more") option tries to take into account automatic wrapping of long lines. However, the code may fail to detect the correct wrapping locations. First, TAB characters (and similar control sequences) are not taken into account, they are handled as ordinary printable characters. Second, depending on the actual system / OS port, *unzip* may not detect the true screen geometry but rather rely on "commonly used" default dimensions. The correct handling of tabs would require the implementation of a query for the actual tabulator setup on the output console.

Dates, times and permissions of stored directories are not restored except under Unix. (On Windows NT and successors, timestamps are now restored.)

[MS-DOS] When extracting or testing files from an archive on a defective floppy diskette, if the "Fail" option is chosen from DOS's "Abort, Retry, Fail?" message, older versions of *unzip* may hang the system, requiring a reboot. This problem appears to be fixed, but control-C (or control-Break) can still be used to terminate *unzip*.

Under DEC Ultrix, *unzip* would sometimes fail on long zipfile (bad CRC, not always reproducible). This was apparently due either to a hardware bug (cache memory) or an operating system bug (improper handling of page faults?). Since Ultrix has been abandoned in favor of Digital Unix (OSF/1), this may not be an issue anymore.

[Unix] Unix special files such as FIFO buffers (named pipes), block devices and character devices are not restored even if they are somehow represented in the zipfile, nor are hard-linked files relinked. Basically the only file types restored by *unzip* are regular files, directories and symbolic (soft) links.

[OS/2] Extended attributes for existing directories are only updated if the **-o** ("overwrite all") option is given. This is a limitation of the operating system; because directories only have a creation time associated with them, *unzip* has no way to determine whether the stored attributes are newer or older than those on disk. In practice this may mean a two-pass approach is required: first unpack the archive normally (with or without freshening/updating existing files), then overwrite just the directory entries (e.g., "*unzip -o foo */*").

[VMS] When extracting to another directory, only the *[foo]* syntax is accepted for the **-d** option; the simple Unix *foo* syntax is silently ignored (as is the less common VMS *foo.dir* syntax).

[VMS] When the file being extracted already exists, *unzip*'s query only allows skipping, overwriting or renaming; there should additionally be a choice for creating a new version of the file. In fact, the "overwrite" choice does create a new version; the old version is not overwritten or deleted.

SEE ALSO

funzip(1), *zip(1)*, *zipcloak(1)*, *zipgrep(1)*, *zipinfo(1)*, *zipnote(1)*, *zipsplit(1)*

URL

The Info-ZIP home page is currently at
<http://www.info-zip.org/pub/infozip/>
 or
<ftp://ftp.info-zip.org/pub/infozip/> .

AUTHORS

The primary Info-ZIP authors (current semi-active members of the Zip-Bugs workgroup) are: Ed Gordon (Zip, general maintenance, shared code, Zip64, Win32, Unix, Unicode); Christian Spieler (UnZip maintenance coordination, VMS, MS-DOS, Win32, shared code, general Zip and UnZip integration and optimization); Onno van der Linden (Zip); Mike White (Win32, Windows GUI, Windows DLLs); Kai Uwe Rommel (OS/2, Win32); Steven M. Schweda (VMS, Unix, support of new features); Paul Kienitz (Amiga, Win32, Unicode); Chris Herborth (BeOS, QNX, Atari); Jonathan Hudson (SMS/QDOS); Sergio Monesi (Acorn RISC OS); Harald Denker (Atari, MVS); John Bush (Solaris, Amiga); Hunter Goatley (VMS, Info-ZIP Site maintenance); Steve Salisbury (Win32); Steve Miller (Windows CE GUI), Johnny Lee (MS-DOS, Win32, Zip64); and Dave Smith (Tandem NSK).

The following people were former members of the Info-ZIP development group and provided major contributions to key parts of the current code: Greg "Cave Newt" Roelofs (UnZip, unshrink decompression); Jean-loup Gailly (deflate compression); Mark Adler (inflate decompression, fUnZip).

The author of the original unzip code upon which Info-ZIP's was based is Samuel H. Smith; Carl Mascott did the first Unix port; and David P. Kirschbaum organized and led Info-ZIP in its early days with Keith Petersen hosting the original mailing list at WSMR-SimTel20. The full list of contributors to UnZip has grown quite large; please refer to the CONTRIBS file in the UnZip source distribution for a relatively complete version.

VERSIONS

v1.2	15 Mar 89	Samuel H. Smith
v2.0	9 Sep 89	Samuel H. Smith
v2.x	fall 1989	many Usenet contributors
v3.0	1 May 90	Info-ZIP (DPK, consolidator)
v3.1	15 Aug 90	Info-ZIP (DPK, consolidator)
v4.0	1 Dec 90	Info-ZIP (GRR, maintainer)
v4.1	12 May 91	Info-ZIP
v4.2	20 Mar 92	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.0	21 Aug 92	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.01	15 Jan 93	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.1	7 Feb 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.11	2 Aug 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.12	28 Aug 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.2	30 Apr 96	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.3	22 Apr 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.31	31 May 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.32	3 Nov 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.4	28 Nov 98	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.41	16 Apr 00	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.42	14 Jan 01	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.5	17 Feb 02	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.51	22 May 04	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.52	28 Feb 05	Info-ZIP (Zip-Bugs subgroup, SPC)
v6.0	20 Apr 09	Info-ZIP (Zip-Bugs subgroup, SPC)

NAME

unzipsfx – self-extracting stub for prepending to ZIP archives

SYNOPSIS

<name of unzipsfx+archive combo> [**-cfptuz[ajnoqsCLV\$]**] [*file(s)* ... [**-x xfile(s)** ...]]

DESCRIPTION

unzipsfx is a modified version of *unzip(1)* designed to be prepended to existing ZIP archives in order to form self-extracting archives. Instead of taking its first non-flag argument to be the zipfile(s) to be extracted, *unzipsfx* seeks itself under the name by which it was invoked and tests or extracts the contents of the appended archive. Because the executable stub adds bulk to the archive (the whole purpose of which is to be as small as possible), a number of the less-vital capabilities in regular *unzip* have been removed. Among these are the usage (or help) screen, the listing and diagnostic functions (**-l** and **-v**), the ability to decompress older compression formats (the “reduce,” “shrink” and “implode” methods). The ability to extract to a directory other than the current one can be selected as a compile-time option, which is now enabled by default since UnZipSFX version 5.5. Similarly, decryption is supported as a compile-time option but should be avoided unless the attached archive contains encrypted files. Starting with release 5.5, another compile-time option adds a simple “run command after extraction” feature. This feature is currently incompatible with the “extract to different directory” feature and remains disabled by default.

Note that self-extracting archives made with unzipsfx are no more (or less) portable across different operating systems than is the unzip executable itself. In general a self-extracting archive made on a particular Unix system, for example, will only self-extract under the same flavor of Unix. Regular *unzip* may still be used to extract the embedded archive as with any normal zipfile, although it will generate a harmless warning about extra bytes at the beginning of the zipfile. *Despite this*, however, the self-extracting archive is technically *not* a valid ZIP archive, and PKUNZIP may be unable to test or extract it. This limitation is due to the simplistic manner in which the archive is created; the internal directory structure is not updated to reflect the extra bytes prepended to the original zipfile.

ARGUMENTS

[*file(s)*] An optional list of archive members to be processed. Regular expressions (wildcards) similar to those in Unix *egrep(1)* may be used to match multiple members. These wildcards may contain:

- * matches a sequence of 0 or more characters
- ? matches exactly 1 character
- [...] matches any single character found inside the brackets; ranges are specified by a beginning character, a hyphen, and an ending character. If an exclamation point or a caret ('!' or '^') follows the left bracket, then the range of characters within the brackets is complemented (that is, anything *except* the characters inside the brackets is considered a match).

(Be sure to quote any character that might otherwise be interpreted or modified by the operating system, particularly under Unix and VMS.)

[**-x xfile(s)**]

An optional list of archive members to be excluded from processing. Since wildcard characters match directory separators ('/'), this option may be used to exclude any files that are in subdirectories. For example, “*foosfx *.[ch] -x */**” would extract all C source files in the main directory, but none in any subdirectories. Without the **-x** option, all C source files in all directories within the zipfile would be extracted.

If *unzipsfx* is compiled with SFX_EXDIR defined, the following option is also enabled:

[**-d exdir**]

An optional directory to which to extract files. By default, all files and subdirectories are recreated in the current directory; the **-d** option allows extraction in an arbitrary directory (always assuming one has permission to write to the directory). The option and directory may be concatenated without any white space between them, but note that this may cause normal shell behavior to be suppressed. In particular, “**-d ~**” (tilde) is expanded by Unix C shells into the name of the user’s home directory, but “**-d ~**” is treated as a literal subdirectory “**~**” of the current directory.

OPTIONS

unzipsfx supports the following *unzip(1)* options: **-c** and **-p** (e xtract to standard output/screen), **-f** and **-u** (freshen and update existing files upon extraction), **-t** (test archive) and **-z** (print archive comment). All normal listing options (**-l**, **-v** and **-Z**) have been removed, but the testing option (**-t**) may be used as a “poor man’s” listing. Alternatively, those creating self-extracting archives may wish to include a short listing in the zipfile comment.

See *unzip(1)* for a more complete description of these options.

MODIFIERS

unzipsfx currently supports all *unzip(1)* modifiers: **-a** (con vert text files), **-n** (never overwrite), **-o** (over-write without prompting), **-q** (operate quietly), **-C** (match names case-insensitively), **-L** (convert uppercase-OS names to lowercase), **-j** (junk paths) and **-V** (retain version numbers); plus the following operating-system specific options: **-X** (restore VMS o wner/protection info), **-s** (convert spaces in filenames to underscores [DOS, OS/2, NT]) and **-\$** (restore volume label [DOS, OS/2, NT, Amiga]).

(Support for regular ASCII text-conversion may be removed in future versions, since it is simple enough for the archive’s creator to ensure that text files have the appropriate format for the local OS. EBCDIC conversion will of course continue to be supported since the zipfile format implies ASCII storage of text files.)

See *unzip(1)* for a more complete description of these modifiers.

ENVIRONMENT OPTIONS

unzipsfx uses the same environment variables as *unzip(1)* does, although this is likely to be an issue only for the person creating and testing the self-extracting archive. See *unzip(1)* for details.

DECRYPTION

Decryption is supported exactly as in *unzip(1)*; that is, interactively with a non-echoing prompt for the password(s). See *unzip(1)* for details. Once again, note that if the archive has no encrypted files there is no reason to use a version of *unzipsfx* with decryption support; that only adds to the size of the archive.

AUTORUN COMMAND

When *unzipsfx* was compiled with **CHEAP_SFX_AUTORUN** defined, a simple “command autorun” feature is supported. You may enter a command into the Zip archive comment, using the following format:

```
$AUTORUN$> [ command line string ]
```

When *unzipsfx* recognizes the “\$AUTORUN\$>” token at the beginning of the Zip archive comment, the remainder of the first line of the comment (until the first newline character) is passed as a shell command to the operating system using the C rtl “system” function. Before executing the command, *unzipsfx* displays the command on the console and prompts the user for confirmation. When the user has switched off prompting by specifying the **-q** option, autorun commands are never executed.

In case the archive comment contains additional lines of text, the remainder of the archive comment following the first line is displayed normally, unless quiet operation was requested by supplying a **-q** option.

EXAMPLES

To create a self-extracting archive *letters* from a regular zipfile *letters.zip* and change the new archive’s permissions to be world-executable under Unix:

```
cat unzipsfx letters.zip > letters
chmod 755 letters
zip -A letters
```

To create the same archive under MS-DOS, OS/2 or NT (note the use of the **/b** [binary] option to the *copy* command):

```
copy /b unzipsfx.exe+letters.zip letters.exe
zip -A letters.exe
```

Under VMS:

```
copy unzipsfx.exe,letters.zip letters.exe
letters == "$currentdisk:[currentdir]letters.exe"
```

```
zip -A letters.exe
```

(The VMS *append* command may also be used. The second command installs the new program as a “foreign command” capable of taking arguments. The third line assumes that Zip is already installed as a foreign command.) Under AmigaDOS:

```
MakeSFX letters letters.zip UnZipSFX
```

(MakeSFX is included with the UnZip source distribution and with Amiga binary distributions. “zip -A” doesn’t work on Amiga self-extracting archives.) To test (or list) the newly created self-extracting archive:

```
letters -t
```

To test *letters* quietly, printing only a summary message indicating whether the archive is OK or not:

```
letters -tqq
```

To extract the complete contents into the current directory, recreating all files and subdirectories as necessary:

```
letters
```

To extract all *.txt files (in Unix quote the ‘*’):

```
letters *.txt
```

To extract everything *except* the *.txt files:

```
letters -x *.txt
```

To extract only the README file to standard output (the screen):

```
letters -c README
```

To print only the zipfile comment:

```
letters -z
```

LIMITATIONS

The principle and fundamental limitation of *unzipsfx* is that it is not portable across architectures or operating systems, and therefore neither are the resulting archives. For some architectures there is limited portability, however (e.g., between some flavors of Intel-based Unix).

Another problem with the current implementation is that any archive with “junk” prepended to the beginning technically is no longer a zipfile (unless *zip(1)* is used to adjust the zipfile offsets appropriately, as noted above). *unzip(1)* takes note of the prepended bytes and ignores them since some file-transfer protocols, notably MacBinary, are also known to prepend junk. But PKWARE’s archiver suite may not be able to deal with the modified archive unless its offsets have been adjusted.

unzipsfx has no knowledge of the user’s PATH, so in general an archive must either be in the current directory when it is invoked, or else a full or relative path must be given. If a user attempts to extract the archive from a directory in the PATH other than the current one, *unzipsfx* will print a warning to the effect, “can’t find myself.” This is always true under Unix and may be true in some cases under MS-DOS, depending on the compiler used (Microsoft C fully qualifies the program name, but other compilers may not). Under OS/2 and NT there are operating-system calls available that provide the full path name, so the archive may be invoked from anywhere in the user’s path. The situation is not known for AmigaDOS, Atari TOS, MacOS, etc.

As noted above, a number of the normal *unzip(1)* functions have been removed in order to make *unzipsfx* smaller: usage and diagnostic info, listing functions and extraction to other directories. Also, only stored and deflated files are supported. The latter limitation is mainly relevant to those who create SFX archives, however.

VMS users must know how to set up self-extracting archives as foreign commands in order to use any of *unzipsfx*’s options. This is not necessary for simple extraction, but the command to do so then becomes, e.g., “run *letters*” (to continue the examples given above).

unzipsfx on the Amiga requires the use of a special program, MakeSFX, in order to create working self-extracting archives; simple concatenation does not work. (For technically oriented users, the attached archive is defined as a “debug hunk.”) There may be compatibility problems between the ROM levels of older Amigas and newer ones.

All current bugs in *unzip(1)* exist in *unzipsfx* as well.

DIAGNOSTICS

unzipsfx’s exit status (error level) is identical to that of *unzip(1)*; see the corresponding man page.

SEE ALSO

funzip(1), *unzip(1)*, *zip(1)*, *zipcloak(1)*, *zipgrep(1)*, *zipinfo(1)*, *zipnote(1)*, *zipsplit(1)*

URL

The Info-ZIP home page is currently at

<http://www.info-zip.org/pub/infozip/>

or

<ftp://ftp.info-zip.org/pub/infozip/> .

AUTHORS

Greg Roelofs was responsible for the basic modifications to UnZip necessary to create UnZipSFX. See *unzip(1)* for the current list of Zip-Bugs authors, or the file CONTRIBS in the UnZip source distribution for the full list of Info-ZIP contributors.

NAME

update-grub, **update-grub2** – stub for grub-mkconfig

SYNOPSIS

update-grub

DESCRIPTION

update-grub is a stub for running **grub-mkconfig -o /boot/grub/grub.cfg** to generate a grub2 config file.

SEE ALSO

grub-mkconfig(8)

NAME

update-grub, **update-grub2** – stub for grub-mkconfig

SYNOPSIS

update-grub

DESCRIPTION

update-grub is a stub for running **grub-mkconfig -o /boot/grub/grub.cfg** to generate a grub2 config file.

SEE ALSO

grub-mkconfig(8)

NAME

updatedb – update a database for plocate

SYNOPSIS

updatedb [*OPTION*]...

DESCRIPTION

updatedb creates or updates a database used by **locate(1)**. If the database already exists, its data is reused to avoid rereading directories that have not changed.

updatedb is usually run daily from a **systemd.timer(8)** to update the default database.

EXIT STATUS

updatedb returns with exit status 0 on success, 1 on error.

OPTIONS

The **PRUNE_BIND_MOUNTS**, **PRUNEFS**, **PRUNENAMES** and **PRUNEPATHS** variables, which are modified by some of the options, are documented in detail in **updatedb.conf(5)**.

-f, --add-prunefs *FS*

Add entries in white-space-separated list *FS* to **PRUNEFS**.

-n, --add-prunenames *NAMES*

Add entries in white-space-separated list *NAMES* to **PRUNENAMES**.

-e, --add-prunepaths *PATHS*

Add entries in white-space-separated list *PATHS* to **PRUNEPATHS**.

-e, --add-single-prunepath *PATH*

Add *PATH* to **PRUNEPATHS**. Note that this is currently the only way to add a path with a space in it.

-U, --database-root *PATH*

Store only results of scanning the file system subtree rooted at *PATH* to the generated database. The whole file system is scanned by default.

locate(1) outputs entries as absolute path names which don't contain symbolic links, regardless of the form of *PATH*.

--debug-pruning

Write debugging information about pruning decisions to standard error output.

-h, --help

Write a summary of the available options to standard output and exit successfully.

-o, --output *FILE*

Write the database to *FILE* instead of using the default database.

--prune-bind-mounts *FLAG*

Set **PRUNE_BIND_MOUNTS** to *FLAG*, overriding the configuration file.

--prunefs *FS*

Set **PRUNEFS** to *FS*, overriding the configuration file.

--prunenames *NAMES*

Set **PRUNENAMES** to *NAMES*, overriding the configuration file.

--prunepaths *PATHS*

Set **PRUNEPATHS** to *PATHS*, overriding the configuration file.

-l, --require-visibility *FLAG*

Set the "require file visibility before reporting it" flag in the generated database to *FLAG*.

If *FLAG* is **0** or **no**, or if the database file is readable by "others" or it is not owned by **plocate**, **locate**(1) outputs the database entries even if the user running **located**(1) could not have read the directory necessary to find out the file described by the database entry.

If *FLAG* is **1** or **yes** (the default), **locate**(1) checks the permissions of parent directories of each entry before reporting it to the invoking user. To make the file existence truly hidden from other users, the database group is set to **plocate** and the database permissions prohibit reading the database by users using other means than **locate**(1), which is set-gid **plocate**.

Note that the visibility flag is checked only if the database is owned by **plocate** and it is not readable by "others".

-v, --verbose

Output path names of files to standard output, as soon as they are found.

-V, --version

Write information about the version and license of **locate** on standard output and exit successfully.

EXAMPLES

To create a private plocate database as a user other than **root**, run

updatedb -l 0 -o db_file -U source_directory

Note that all users that can read *db_file* can get the complete list of files in the subtree of *source_directory*.

FILES

/etc/updatedb.conf

A configuration file. See **updatedb.conf**(5). Uses exactly the same format as the one used by **mlocate**(1)'s updatedb, so they can be shared.

/var/lib/plocate/plocate.db

The database updated by default.

SECURITY

Databases built with **--require-visibility no** allow users to find names of files and directories of other users, which they would not otherwise be able to do.

AUTHOR

Miloslav Trmac <mitr@redhat.com>

Steinar H. Gunderson <steinar+plocate@gunderson.no>

SEE ALSO

locate(1), updatedb.conf(5)

NAME

uptime – Tell how long the system has been running.

SYNOPSIS

uptime [*options*]

DESCRIPTION

uptime gives a one line display of the following information. The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

This is the same information contained in the header line displayed by **w(1)**.

System load averages is the average number of processes that are either in a runnable or uninterruptable state. A process in a runnable state is either using the CPU or waiting to use the CPU. A process in uninterruptable state is waiting for some I/O access, eg waiting for disk. The averages are taken over the three time intervals. Load averages are not normalized for the number of CPUs in a system, so a load average of 1 means a single CPU system is loaded all the time while on a 4 CPU system it means it was idle 75% of the time.

OPTIONS

-p, --pretty

show uptime in pretty format

-h, --help

display this help text

-s, --since

system up since, in yyyy-mm-dd HH:MM:SS format

-V, --version

display version information and exit

FILES

/var/run/utmp

information about who is currently logged on

/proc process information

AUTHORS

uptime was written by Larry Greenfield <greenfie@gauss.rutgers.edu> and Michael K. Johnson <johnsonm@sunsite.unc.edu>

SEE ALSO

ps(1), top(1), utmp(5), w(1)

REPORTING BUGS

Please send bug reports to <procps@freelists.org>

NAME

useradd – create a new user or update default new user information

SYNOPSIS

useradd [*options*] *LOGIN*

useradd –D

useradd –D [*options*]

DESCRIPTION

useradd is a low level utility for adding users. On Debian, administrators should usually use **adduser(8)** instead.

When invoked without the **–D** option, the **useradd** command creates a new user account using the values specified on the command line plus the default values from the system. Depending on command line options, the **useradd** command will update system files and may also create the new user's home directory and copy initial files.

By default, a group will also be created for the new user (see **–g**, **–N**, **–U**, and **USERGROUPS_ENAB**).

OPTIONS

The options which apply to the **useradd** command are:

--badname

Allow names that do not conform to standards.

–b, --base-dir *BASE_DIR*

The default base directory for the system if **–d HOME_DIR** is not specified. *BASE_DIR* is concatenated with the account name to define the home directory. If the **–m** option is not used, *BASE_DIR* must exist.

If this option is not specified, **useradd** will use the base directory specified by the **HOME** variable in /etc/default/useradd, or /home by default.

–c, --comment *COMMENT*

Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name.

–d, --home-dir *HOME_DIR*

The new user will be created using *HOME_DIR* as the value for the user's login directory. The default is to append the *LOGIN* name to *BASE_DIR* and use that as the login directory name. The directory *HOME_DIR* does not have to exist but will not be created if it is missing.

–D, --defaults

See below, the subsection "Changing the default values".

–e, --expiredate *EXPIRE_DATE*

The date on which the user account will be disabled. The date is specified in the format *YYYY-MM-DD*.

If not specified, **useradd** will use the default expiry date specified by the **EXPIRE** variable in /etc/default/useradd, or an empty string (no expiry) by default.

–f, --inactive *INACTIVE*

The number of days after a password expires until the account is permanently disabled. A value of 0 disables the account as soon as the password has expired, and a value of –1 disables the feature.

If not specified, **useradd** will use the default inactivity period specified by the **INACTIVE** variable in /etc/default/useradd, or –1 by default.

–g, --gid *GROUP*

The group name or number of the user's initial login group. The group name must exist. A group

number must refer to an already existing group.

If not specified, the behavior of **useradd** will depend on the **USERGROUPS_ENAB** variable in /etc/login.defs. If this variable is set to *yes* (or **-U**/**--user-group** is specified on the command line), a group will be created for the user, with the same name as her loginname. If the variable is set to *no* (or **-N**/**--no-user-group** is specified on the command line), useradd will set the primary group of the new user to the value specified by the **GROUP** variable in /etc/default/useradd, or 100 by default.

-G, --groups GROUP1[,GROUP2,...,[GROUPN]]

A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the **-g** option. The default is for the user to belong only to the initial group.

-h, --help

Display help message and exit.

-k, --skel SKEL_DIR

The skeleton directory, which contains files and directories to be copied in the user's home directory, when the home directory is created by **useradd**.

This option is only valid if the **-m** (or **--create-home**) option is specified.

If this option is not set, the skeleton directory is defined by the **SKEL** variable in /etc/default/useradd or, by default, /etc/skel.

If possible, the ACLs and extended attributes are copied.

-K, --key KEY=VALUE

Overrides /etc/login.defs defaults (**UID_MIN**, **UID_MAX**, **UMASK**, **PASS_MAX_DAYS** and others).

Example: **-K PASS_MAX_DAYS=-1** can be used when creating system account to turn off password aging, even though system account has no password at all. Multiple **-K** options can be specified, e.g.:
-K UID_MIN=100 -K UID_MAX=499

-l, --no-log-init

Do not add the user to the lastlog and faillog databases.

By default, the user's entries in the lastlog and faillog databases are reset to avoid reusing the entry from a previously deleted user.

For the compatibility with previous Debian's **useradd**, the **-O** option is also supported.

-m, --create-home

Create the user's home directory if it does not exist. The files and directories contained in the skeleton directory (which can be defined with the **-k** option) will be copied to the home directory.

By default, if this option is not specified and **CREATE_HOME** is not enabled, no home directories are created.

-M, --no-create-home

Do not create the user's home directory, even if the system wide setting from /etc/login.defs (**CREATE_HOME**) is set to *yes*.

-N, --no-user-group

Do not create a group with the same name as the user, but add the user to the group specified by the **-g** option or by the **GROUP** variable in /etc/default/useradd.

The default behavior (if the **-g**, **-N**, and **-U** options are not specified) is defined by the

USERGROUPS_ENAB variable in /etc/login.defs.

-o, --non-unique

Allow the creation of a user account with a duplicate (non-unique) UID.

This option is only valid in combination with the **-u** option.

-p, --password *PASSWORD*

The encrypted password, as returned by **crypt(3)**. The default is to disable the password.

Note: This option is not recommended because the password (or encrypted password) will be visible by users listing the processes.

You should make sure the password respects the system's password policy.

-r, --system

Create a system account.

System users will be created with no aging information in /etc/shadow, and their numeric identifiers are chosen in the **SYS_UID_MIN**–**SYS_UID_MAX** range, defined in /etc/login.defs, instead of **UID_MIN**–**UID_MAX** (and their **GID** counterparts for the creation of groups).

Note that **useradd** will not create a home directory for such a user, regardless of the default setting in /etc/login.defs (**CREATE_HOME**). You have to specify the **-m** options if you want a home directory for a system account to be created.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix *PREFIX_DIR*

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

-s, --shell *SHELL*

The name of the user's login shell. The default is to leave this field blank, which causes the system to select the default login shell specified by the **SHELL** variable in /etc/default/useradd, or an empty string by default.

-u, --uid *UID*

The numerical value of the user's ID. This value must be unique, unless the **-o** option is used. The value must be non-negative. The default is to use the smallest ID value greater than or equal to **UID_MIN** and greater than every other user.

See also the **-r** option and the **UID_MAX** description.

-U, --user-group

Create a group with the same name as the user, and add the user to this group.

The default behavior (if the **-g**, **-N**, and **-U** options are not specified) is defined by the **USERGROUPS_ENAB** variable in /etc/login.defs.

-Z, --selinux-user *SEUSER*

The SELinux user for the user's login. The default is to leave this field blank, which causes the system to select the default SELinux user.

Changing the default values

When invoked with only the **-D** option, **useradd** will display the current default values. When invoked with **-D** plus other options, **useradd** will update the default values for the specified options. Valid default-changing options are:

-b, --base-dir *BASE_DIR*

The path prefix for a new user's home directory. The user's name will be affixed to the end of *BASE_DIR* to form the new user's home directory name, if the **-d** option is not used when creating a new account.

This option sets the **HOME** variable in /etc/default/useradd.

-e, --expiredate *EXPIRE_DATE*

The date on which the user account is disabled.

This option sets the **EXPIRE** variable in /etc/default/useradd.

-f, --inactive *INACTIVE*

The number of days after a password has expired before the account will be disabled.

This option sets the **INACTIVE** variable in /etc/default/useradd.

-g, --gid *GROUP*

The group name or ID for a new user's initial group (when the **-N**/**--no-user-group** is used or when the **USERGROUPS_ENAB** variable is set to *no* in /etc/login.defs). The named group must exist, and a numerical group ID must have an existing entry.

This option sets the **GROUP** variable in /etc/default/useradd.

-s, --shell *SHELL*

The name of a new user's login shell.

This option sets the **SHELL** variable in /etc/default/useradd.

NOTES

The system administrator is responsible for placing the default user files in the /etc/skel/ directory (or any other skeleton directory specified in /etc/default/useradd or on the command line).

CAVEATS

You may not add a user to a NIS or LDAP group. This must be performed on the corresponding server.

Similarly, if the username already exists in an external user database such as NIS or LDAP, **useradd** will deny the user account creation request.

It is usually recommended to only use usernames that begin with a lower case letter or an underscore, followed by lower case letters, digits, underscores, or dashes. They can end with a dollar sign. In regular expression terms: [a-z_][a-z0-9_-]*[\$]?

On Debian, the only constraints are that usernames must neither start with a dash ('-') nor plus ('+') nor tilde ('~') nor contain a colon (':'), a comma (','), or a whitespace (space: ' ', end of line: '\n', tabulation: '\t', etc.). Note that using a slash ('/') may break the default algorithm for the definition of the user's home directory.

On Ubuntu, the same constraints as Debian are in place, with the additional constraint that the username cannot be fully numeric. This includes octal and hexadecimal syntax.

Usernames may only be up to 32 characters long.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

CREATE_HOME (boolean)

Indicate if a home directory should be created by default for new users.

This setting does not apply to system users, and can be overridden on the command line.

GID_MAX (number), **GID_MIN** (number)

Range of group IDs used for the creation of regular groups by **useradd**, **groupadd**, or **newusers**.

The default value for **GID_MIN** (resp. **GID_MAX**) is 1000 (resp. 60000).

HOME_MODE (number)

The mode for new home directories. If not specified, the **UMASK** is used to create the mode.

useradd and **newusers** use this to set the mode of the home directory they create.

LASTLOG_UID_MAX (number)

Highest user ID number for which the lastlog entries should be updated. As higher user IDs are usually tracked by remote user identity and authentication services there is no need to create a huge sparse lastlog file for them.

No **LASTLOG_UID_MAX** option present in the configuration means that there is no user ID limit for writing lastlog entries.

MAIL_DIR (string)

The mail spool directory. This is needed to manipulate the mailbox when its corresponding user account is modified or deleted. If not specified, a compile-time default is used.

MAIL_FILE (string)

Defines the location of the users mail spool files relatively to their home directory.

The **MAIL_DIR** and **MAIL_FILE** variables are used by **useradd**, **usermod**, and **userdel** to create, move, or delete the user's mail spool.

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

PASS_MAX_DAYS (number)

The maximum number of days a password may be used. If the password is older than this, a password change will be forced. If not specified, -1 will be assumed (which disables the restriction).

PASS_MIN_DAYS (number)

The minimum number of days allowed between password changes. Any password changes attempted sooner than this will be rejected. If not specified, -1 will be assumed (which disables the restriction).

PASS_WARN_AGE (number)

The number of days warning given before a password expires. A zero means warning is given only upon the day of expiration, a negative value means no warning is given. If not specified, no warning will be provided.

SUB_GID_MIN (number), **SUB_GID_MAX** (number), **SUB_GID_COUNT** (number)

If /etc/subuid exists, the commands **useradd** and **newusers** (unless the user already have subordinate group IDs) allocate **SUB_GID_COUNT** unused group IDs from the range **SUB_GID_MIN** to **SUB_GID_MAX** for each new user.

The default values for **SUB_GID_MIN**, **SUB_GID_MAX**, **SUB_GID_COUNT** are respectively 100000, 600100000 and 65536.

SUB_UID_MIN (number), **SUB_UID_MAX** (number), **SUB_UID_COUNT** (number)

If /etc/subuid exists, the commands **useradd** and **newusers** (unless the user already have subordinate user IDs) allocate **SUB_UID_COUNT** unused user IDs from the range **SUB_UID_MIN** to **SUB_UID_MAX** for each new user.

The default values for **SUB_UID_MIN**, **SUB_UID_MAX**, **SUB_UID_COUNT** are respectively 100000, 600100000 and 65536.

SYS_GID_MAX (number), **SYS_GID_MIN** (number)

Range of group IDs used for the creation of system groups by **useradd**, **groupadd**, or **newusers**.

The default value for **SYS_GID_MIN** (resp. **SYS_GID_MAX**) is 101 (resp. **GID_MIN**-1).

SYS_UID_MAX (number), **SYS_UID_MIN** (number)

Range of user IDs used for the creation of system users by **useradd** or **newusers**.

The default value for **SYS_UID_MIN** (resp. **SYS_UID_MAX**) is 101 (resp. **UID_MIN**-1).

UID_MAX (number), **UID_MIN** (number)

Range of user IDs used for the creation of regular users by **useradd** or **newusers**.

The default value for **UID_MIN** (resp. **UID_MAX**) is 1000 (resp. 60000).

UMASK (number)

The file mode creation mask is initialized to this value. If not specified, the mask will be initialized to 022.

useradd and **newusers** use this mask to set the mode of the home directory they create if **HOME_MODE** is not set.

It is also used by **pam_umask** as the default umask value.

USERGROUPS_ENAB (boolean)

If set to **yes**, **userdel** will remove the user's group if it contains no more members, and **useradd** will create by default a group with the name of the user.

FILES

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

/etc/group

Group account information.

/etc/gshadow

Secure group account information.

/etc/default/useradd

Default values for account creation.

/etc/skel/

Directory containing default files.

/etc/subgid

Per user subordinate group IDs.

/etc/subuid

Per user subordinate user IDs.

/etc/login.defs
Shadow password suite configuration.

EXIT VALUES

The **useradd** command exits with the following values:

- 0* success
- 1* can't update password file
- 2* invalid command syntax
- 3* invalid argument to option
- 4* UID already in use (and no **-o**)
- 6* specified group doesn't exist
- 9* username already in use
- 10* can't update group file
- 12* can't create home directory
- 14* can't update SELinux user mapping

SEE ALSO

chfn(1), chsh(1), passwd(1), crypt(3), groupadd(8), groupdel(8), groupmod(8), login.defs(5), newusers(8), subgid(5), subuid(5), userdel(8), usermod(8).

NAME

userdel – delete a user account and related files

SYNOPSIS

userdel [options] *LOGIN*

DESCRIPTION

userdel is a low level utility for removing users. On Debian, administrators should usually use **deluser(8)** instead.

The **userdel** command modifies the system account files, deleting all entries that refer to the user name *LOGIN*. The named user must exist.

OPTIONS

The options which apply to the **userdel** command are:

-f, --force

This option forces the removal of the user account, even if the user is still logged in. It also forces **userdel** to remove the user's home directory and mail spool, even if another user uses the same home directory or if the mail spool is not owned by the specified user. If **USERGROUPS_ENAB** is defined to *yes* in /etc/login.defs and if a group exists with the same name as the deleted user, then this group will be removed, even if it is still the primary group of another user.

Note: This option is dangerous and may leave your system in an inconsistent state.

-h, --help

Display help message and exit.

-r, --remove

Files in the user's home directory will be removed along with the home directory itself and the user's mail spool. Files located in other file systems will have to be searched for and deleted manually.

The mail spool is defined by the **MAIL_DIR** variable in the login.defs file.

-R, --root *CHROOT_DIR*

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix *PREFIX_DIR*

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

-Z, --selinux-user

Remove any SELinux user mapping for the user's login.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

MAIL_DIR (string)

The mail spool directory. This is needed to manipulate the mailbox when its corresponding user account is modified or deleted. If not specified, a compile-time default is used.

MAIL_FILE (string)

Defines the location of the users mail spool files relatively to their home directory.

The **MAIL_DIR** and **MAIL_FILE** variables are used by **useradd**, **usermod**, and **userdel** to create, move, or delete the user's mail spool.

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

USERDEL_CMD (string)

If defined, this command is run when removing a user. It should remove any at/cron/print jobs etc. owned by the user to be removed (passed as the first argument).

The return code of the script is not taken into account.

Here is an example script, which removes the user's cron, at and print jobs:

```
#!/bin/sh

# Check for the required argument.
if [ $# != 1 ]; then
    echo "Usage: $0 username"
    exit 1
fi

# Remove cron jobs.
crontab -r -u $1

# Remove at jobs.
# Note that it will remove any jobs owned by the same UID,
# even if it was shared by a different username.
AT_SPOOL_DIR=/var/spool/cron/atjobs
find $AT_SPOOL_DIR -name "[^.]*" -type f -user $1 -delete \;

# Remove print jobs.
lprm $1

# All done.
exit 0
```

USERGROUPS_ENAB (boolean)

If set to yes, **userdel** will remove the user's group if it contains no more members, and **useradd** will create by default a group with the name of the user.

FILES

/etc/group

Group account information.

/etc/login.defs

Shadow password suite configuration.

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

/etc/subgid

Per user subordinate group IDs.

/etc/subuid

Per user subordinate user IDs.

EXIT VALUES

The **userdel** command exits with the following values:

0

success

1

can't update password file

2

invalid command syntax

6

specified user doesn't exist

8

user currently logged in

10

can't update group file

12

can't remove home directory

CAVEATS

userdel will not allow you to remove an account if there are running processes which belong to this account. In that case, you may have to kill those processes or lock the user's password or account and remove the account later. The **-f** option can force the deletion of this account.

You should manually check all file systems to ensure that no files remain owned by this user.

You may not remove any NIS attributes on a NIS client. This must be performed on the NIS server.

If **USERGROUPS_ENAB** is defined to *yes* in /etc/login.defs, **userdel** will delete the group with the same name as the user. To avoid inconsistencies in the passwd and group databases, **userdel** will check that this group is not used as a primary group for another user, and will just warn without deleting the group otherwise. The **-f** option can force the deletion of this group.

SEE ALSO

chfn(1), **chsh(1)**, **passwd(1)**, **login.defs(5)**, **gpasswd(8)**, **groupadd(8)**, **groupdel(8)**, **groupmod(8)**, **subgid(5)**, **subuid(5)**, **useradd(8)**, **usermod(8)**.

NAME

usermod – modify a user account

SYNOPSIS

usermod [*options*] *LOGIN*

DESCRIPTION

The **usermod** command modifies the system account files to reflect the changes that are specified on the command line.

OPTIONS

The options which apply to the **usermod** command are:

-a, --append

Add the user to the supplementary group(s). Use only with the **-G** option.

-b, --badnames

Allow names that do not conform to standards.

-c, --comment *COMMENT*

The new value of the user's password file comment field. It is normally modified using the **chfn(1)** utility.

-d, --home *HOME_DIR*

The user's new login directory.

If the **-m** option is given, the contents of the current home directory will be moved to the new home directory, which is created if it does not already exist.

-e, --expiredate *EXPIRE_DATE*

The date on which the user account will be disabled. The date is specified in the format *YYYY-MM-DD*.

An empty *EXPIRE_DATE* argument will disable the expiration of the account.

This option requires a /etc/shadow file. A /etc/shadow entry will be created if there were none.

-f, --inactive *INACTIVE*

The number of days after a password expires until the account is permanently disabled.

A value of 0 disables the account as soon as the password has expired, and a value of -1 disables the feature.

This option requires a /etc/shadow file. A /etc/shadow entry will be created if there were none.

-g, --gid *GROUP*

The group name or number of the user's new initial login group. The group must exist.

Any file from the user's home directory owned by the previous primary group of the user will be owned by this new group.

The group ownership of files outside of the user's home directory must be fixed manually.

-G, --groups *GROUP1[,GROUP2,...[,GROUPN]]*

A list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the **-g** option.

If the user is currently a member of a group which is not listed, the user will be removed from the group. This behaviour can be changed via the **-a** option, which appends the user to the current supplementary group list.

-l, --login NEW_LOGIN

The name of the user will be changed from *LOGIN* to *NEW_LOGIN*. Nothing else is changed. In particular, the user's home directory or mail spool should probably be renamed manually to reflect the new login name.

-L, --lock

Lock a user's password. This puts a '!' in front of the encrypted password, effectively disabling the password. You can't use this option with **-p** or **-U**.

Note: if you wish to lock the account (not only access with a password), you should also set the *EXPIRE_DATE* to *1*.

-m, --move-home

Move the content of the user's home directory to the new location.

This option is only valid in combination with the **-d** (or **--home**) option.

usermod will try to adapt the ownership of the files and to copy the modes, ACL and extended attributes, but manual changes might be needed afterwards.

-o, --non-unique

When used with the **-u** option, this option allows to change the user ID to a non-unique value.

-p, --password PASSWORD

The encrypted password, as returned by **crypt(3)**.

Note: This option is not recommended because the password (or encrypted password) will be visible by users listing the processes.

The password will be written in the local /etc/passwd or /etc/shadow file. This might differ from the password database configured in your PAM configuration.

You should make sure the password respects the system's password policy.

-R, --root CHROOT_DIR

Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

-P, --prefix PREFIX_DIR

Apply changes in the *PREFIX_DIR* directory and use the configuration files from the *PREFIX_DIR* directory. This option does not chroot and is intended for preparing a cross-compilation target. Some limitations: NIS and LDAP users/groups are not verified. PAM authentication is using the host files. No SELINUX support.

-s, --shell SHELL

The name of the user's new login shell. Setting this field to blank causes the system to select the default login shell.

-u, --uid UID

The new numerical value of the user's ID.

This value must be unique, unless the **-o** option is used. The value must be non-negative.

The user's mailbox, and any files which the user owns and which are located in the user's home directory will have the file user ID changed automatically.

The ownership of files outside of the user's home directory must be fixed manually.

No checks will be performed with regard to the **UID_MIN**, **UID_MAX**, **SYS_UID_MIN**, or

SYS_UID_MAX from /etc/login.defs.

-U, --unlock

Unlock a user's password. This removes the '!' in front of the encrypted password. You can't use this option with **-p** or **-L**.

Note: if you wish to unlock the account (not only access with a password), you should also set the **EXPIRE_DATE** (for example to 99999, or to the **EXPIRE** value from /etc/default/useradd).

-v, --add-subuids FIRST-LAST

Add a range of subordinate uids to the user's account.

This option may be specified multiple times to add multiple ranges to a users account.

No checks will be performed with regard to **SUB_UID_MIN**, **SUB_UID_MAX**, or **SUB_UID_COUNT** from /etc/login.defs.

-V, --del-subuids FIRST-LAST

Remove a range of subordinate uids from the user's account.

This option may be specified multiple times to remove multiple ranges to a users account. When both **--del-subuids** and **--add-subuids** are specified, the removal of all subordinate uid ranges happens before any subordinate uid range is added.

No checks will be performed with regard to **SUB_UID_MIN**, **SUB_UID_MAX**, or **SUB_UID_COUNT** from /etc/login.defs.

-w, --add-subgids FIRST-LAST

Add a range of subordinate gids to the user's account.

This option may be specified multiple times to add multiple ranges to a users account.

No checks will be performed with regard to **SUB_GID_MIN**, **SUB_GID_MAX**, or **SUB_GID_COUNT** from /etc/login.defs.

-W, --del-subgids FIRST-LAST

Remove a range of subordinate gids from the user's account.

This option may be specified multiple times to remove multiple ranges to a users account. When both **--del-subgids** and **--add-subgids** are specified, the removal of all subordinate gid ranges happens before any subordinate gid range is added.

No checks will be performed with regard to **SUB_GID_MIN**, **SUB_GID_MAX**, or **SUB_GID_COUNT** from /etc/login.defs.

-Z, --selinux-user SEUSER

The new SELinux user for the user's login.

A blank **SEUSER** will remove the SELinux user mapping for user **LOGIN** (if any).

CAVEATS

You must make certain that the named user is not executing any processes when this command is being executed if the user's numerical user ID, the user's name, or the user's home directory is being changed. **usermod** checks this on Linux. On other platforms it only uses utmp to check if the user is logged in.

You must change the owner of any **crontab** files or **at** jobs manually.

You must make any changes involving NIS on the NIS server.

CONFIGURATION

The following configuration variables in /etc/login.defs change the behavior of this tool:

LASTLOG_UID_MAX (number)

Highest user ID number for which the lastlog entries should be updated. As higher user IDs are usually tracked by remote user identity and authentication services there is no need to create a huge sparse lastlog file for them.

No **LASTLOG_UID_MAX** option present in the configuration means that there is no user ID limit for writing lastlog entries.

MAIL_DIR (string)

The mail spool directory. This is needed to manipulate the mailbox when its corresponding user account is modified or deleted. If not specified, a compile-time default is used.

MAIL_FILE (string)

Defines the location of the users mail spool files relatively to their home directory.

The **MAIL_DIR** and **MAIL_FILE** variables are used by **useradd**, **usermod**, and **userdel** to create, move, or delete the user's mail spool.

MAX_MEMBERS_PER_GROUP (number)

Maximum members per group entry. When the maximum is reached, a new group entry (line) is started in /etc/group (with the same name, same password, and same GID).

The default value is 0, meaning that there are no limits in the number of members in a group.

This feature (split group) permits to limit the length of lines in the group file. This is useful to make sure that lines for NIS groups are not larger than 1024 characters.

If you need to enforce such limit, you can use 25.

Note: split groups may not be supported by all tools (even in the Shadow toolsuite). You should not use this variable unless you really need it.

SUB_GID_MIN (number), **SUB_GID_MAX** (number), **SUB_GID_COUNT** (number)

If /etc/subuid exists, the commands **useradd** and **newusers** (unless the user already have subordinate group IDs) allocate **SUB_GID_COUNT** unused group IDs from the range **SUB_GID_MIN** to **SUB_GID_MAX** for each new user.

The default values for **SUB_GID_MIN**, **SUB_GID_MAX**, **SUB_GID_COUNT** are respectively 100000, 600100000 and 65536.

SUB_UID_MIN (number), **SUB_UID_MAX** (number), **SUB_UID_COUNT** (number)

If /etc/subuid exists, the commands **useradd** and **newusers** (unless the user already have subordinate user IDs) allocate **SUB_UID_COUNT** unused user IDs from the range **SUB_UID_MIN** to **SUB_UID_MAX** for each new user.

The default values for **SUB_UID_MIN**, **SUB_UID_MAX**, **SUB_UID_COUNT** are respectively 100000, 600100000 and 65536.

FILES

/etc/group

Group account information.

/etc/gshadow

Secure group account information.

/etc/login.defs

Shadow password suite configuration.

/etc/passwd
User account information.

/etc/shadow
Secure user account information.

/etc/subgid
Per user subordinate group IDs.

/etc/subuid
Per user subordinate user IDs.

SEE ALSO

chfn(1), chsh(1), passwd(1), crypt(3), gpasswd(8), groupadd(8), groupdel(8), groupmod(8), login.defs(5), subgid(5), subuid(5), useradd(8), userdel(8).

NAME

users – print the user names of users currently logged in to the current host

SYNOPSIS

users [*OPTION*]... [*FILE*]

DESCRIPTION

Output who is currently logged in according to FILE. If FILE is not specified, use */var/run/utmp*. */var/log/wtmp* as FILE is common.

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by Joseph Arceneaux and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

getent(1), **who(1)**

Full documentation <<https://www.gnu.org/software/coreutils/users>>
or available locally via: info '(coreutils) users invocation'

NAME

`vgcfgbackup` – Backup volume group configuration(s)

SYNOPSIS

```
vgcfgbackup
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

`vgcfgbackup` creates back up files containing metadata of VGs. If no VGs are named, back up files are created for all VGs. See `vgcfgrestore` for information on using the back up files.

In a default installation, each VG is backed up into a separate file bearing the name of the VG in the directory `/etc/lvm/backup`.

To use an alternative back up file, use `-f`. In this case, when backing up multiple VGs, the file name is treated as a template, with %s replaced by the VG name.

NB. This DOES NOT back up the data content of LVs.

It may also be useful to regularly back up the files in `/etc/lvm`.

USAGE

```
vgcfgbackup
  [ -f|--file String ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ COMMON_OPTIONS ]
  [ VG ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See `lvm.conf(5)` for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See `lvm.conf(5)` for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--file String

Write the backup to the named file. When backing up more than one VG, the file name is treated as a template, and %s is replaced by the VG name.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme

caution. (For automatic no, see `-qq`.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgcfgrestore – Restore volume group configuration

SYNOPSIS

```
vgcfgrestore option_args position_args
  [ option_args ]
  [ position_args ]

  --commandprofile String
  --config String
  -d|--debug
  --driverloaded y|n
  -f|--file String
  --force
  -h|--help
  -l|--list
  --lockopt String
  --longhelp
  -M|--metadatatype lvm2
  --nolocking
  --profile String
  -q|--quiet
  -t|--test
  -v|--verbose
  --version
  -y|--yes
```

DESCRIPTION

vgcfgrestore restores the metadata of a VG from a text back up file produced by **vgcfgbackup**. This writes VG metadata onto the devices specified in back up file.

A back up file can be specified with **--file**. If no backup file is specified, the most recent one is used. Use **--list** for a list of the available back up and archive files of a VG.

WARNING: When a VG contains thin pools, changes to thin metadata cannot be reverted, and data loss may occur if thin metadata has changed. The force option is required to restore in this case.

USAGE

Restore VG metadata from last backup.

```
vgcfgrestore VG
  [ COMMON_OPTIONS ]
```

Restore VG metadata from specified file.

```
vgcfgrestore -f|--file String VG
  [ COMMON_OPTIONS ]
```

List all VG metadata backups.

```
vgcfgrestore -l|--list VG
  [ COMMON_OPTIONS ]
```

List one VG metadata backup file.

```
vgcfgrestore -l|--list -f|--file String
```

```
[ COMMON_OPTIONS ]
[ VG ]
```

Common options for command:

```
[ -M|--metadatatype lvm2 ]
[ --force ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--file *String*

Read metadata backup from the named file. Usually this file was created by vgcfgbackup.

--force ...

Force metadata restore even with thin pool LVs. Use with extreme caution. Most changes to thin metadata cannot be reverted. You may lose data if you restore metadata that does not match the thin pool kernel metadata precisely.

-h|--help

Display help text.

-l|--list

List metadata backup and archive files pertaining to the VG. May be used with --file. Does not restore the VG.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-M|--metadatatype lvm2

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format. **lvm1** (or just **1**) is no longer used.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

To replace PVs, **vgdisplay --partial --verbose** will show the UUIDs and sizes of any PVs that are no longer present. If a PV in the VG is lost and you wish to substitute another of the same size, use **pvcreate --restorefile filename --uuid uuid** (plus additional arguments as appropriate) to initialise it with the same UUID as the missing PV. Repeat for all other missing PVs in the VG. Then use **vgcfgrestore --file filename** to restore the VG's metadata.

SEE ALSO

lvm(8) **lvm.conf(5)** **lvmconfig(8)**

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)
vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)
lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)
lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgchange – Change volume group attributes

SYNOPSIS

```
vgchange option_args position_args
  [ option_args ]
  [ position_args ]

-a|--activate y|n|ay
  --activationmode partial|degraded|complete
  --addtag Tag
  --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit
-A|--autobackup y|n
  --commandprofile String
  --config String
-d|--debug
  --deltag Tag
  --detachprofile
  --driverloaded y|n
-f|--force
-h|--help
-K|--ignoreactivationskip
  --ignorelockingfailure
  --ignoremonitoring
  --lockopt String
  --lockstart
  --lockstop
  --locktype sanlock|dlm|none
-l|--logicalvolume Number
  --longhelp
-p|--maxphysicalvolumes Number
  --metadaprofile String
  --monitor y|n
  --nolocking
  --noudevsync
-P|--partial
-s|--physicalextentsize Size[m|UNIT]
  --poll y|n
  --profile String
  --pvmetadatacopies 0|1|2
-q|--quiet
  --readonly
  --refresh
  --reportformat basic|json
-x|--resizeable y|n
-S|--select String
  --sysinit
  --systemid String
-t|--test
-u|--uuid
-v|--verbose
  --version
  --[vg]metadatacopies all|unmanaged|Number
-y|--yes
```

DESCRIPTION

`vgchange` changes VG attributes, changes LV activation in the kernel, and includes other utilities for VG maintenance.

USAGE

Change a general VG attribute.

For options listed in parentheses, any one is required, after which the others are optional.

vgchange

```
( -l|--logicalvolume Number,
  -p|--maxphysicalvolumes Number,
  -u|--uuid,
  -s|--physicalextentsize Size[m|UNIT],
  -x|--resizeable y|n,
  --addtag Tag,
  --deltag Tag,
  --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit,
  --pvmetadatacopies 0|1|2,
  --[vg]metadatacopies all|unmanaged|Number,
  --profile String,
  --detachprofile,
  --metadataprofile String)
[ -A|--autobackup y|n ]
[ -S|--select String ]
[ -f|--force ]
[ --poll y|n ]
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

Start or stop monitoring LVs from dmeventd.

vgchange ---monitor y|n

```
[ -A|--autobackup y|n ]
[ -S|--select String ]
[ -f|--force ]
[ --sysinit ]
[ --ignorelockingfailure ]
[ --poll y|n ]
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

Start or stop processing LV conversions.

vgchange ---poll y|n

```
[ -A|--autobackup y|n ]
[ -S|--select String ]
[ -f|--force ]
[ --ignorelockingfailure ]
```

```
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

-
Activate or deactivate LVs.

```
vgchange -a|--activate y|n|ay
[ -K|--ignoreactivationskip ]
[ -P|--partial ]
[ -A|--autobackup y|n ]
[ -S|--select String ]
[ -f|--force ]
[ --activationmode partial|degraded|complete ]
[ --sysinit ]
[ --readonly ]
[ --ignorelockingfailure ]
[ --monitor y|n ]
[ --poll y|n ]
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

-
Reactivate LVs using the latest metadata.

```
vgchange --refresh
[ -A|--autobackup y|n ]
[ -S|--select String ]
[ -f|--force ]
[ --sysinit ]
[ --ignorelockingfailure ]
[ --poll y|n ]
[ --ignoremonitoring ]
[ --noudevsync ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

-
Change the system ID of a VG.

```
vgchange --systemid String VG
[ COMMON_OPTIONS ]
```

-
Start the lockspace of a shared VG in lvmlockd.

```
vgchange --lockstart
[ -S|--select String ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

Stop the lockspace of a shared VG in lvmlockd.

```
vgchange --lockstop
[ -S|--select String ]
[ COMMON_OPTIONS ]
[ VG|Tag|Select ... ]
```

Change the lock type for a shared VG.

```
vgchange --locktype sanlock|dlm|none VG
[ COMMON_OPTIONS ]
```

Common options for command:

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-a|--activate y|n|ay

Change the active state of LVs. An active LV can be used through a block device, allowing data on the LV to be accessed. **y** makes LVs active, or available. **n** makes LVs inactive, or unavailable. The block device for the LV is added or removed from the system using device-mapper in the kernel. A symbolic link /dev/VGName/LVName pointing to the device node is also added/removed. All software and scripts should access the device through the symbolic link and present this as the name of the device. The location and name of the underlying device node may depend on the distribution, configuration (e.g. udev), or release version. **ay** specifies autoactivation, in which case an LV is activated only if it matches an item in lvm.conf activation/auto_activation_volume_list. If the list is not set, all LVs are considered to match, and if the list is set but empty, no LVs match. Autoactivation should be used during system boot to make it possible to select which LVs should be automatically activated by the system. See **lvmlockd(8)** for more information about activation options **ey** and **sy** for shared VGs.

--activationmode partial|degraded|complete

Determines if LV activation is allowed when PVs are missing, e.g. because of a device failure. **complete** only allows LVs with no missing PVs to be activated, and is the most restrictive mode. **degraded** allows RAID LVs with missing PVs to be activated. (This does not include the "mirror" type, see "raid1" instead.) **partial** allows any LV with missing PVs to be activated, and should only be used for recovery or repair. For default, see lvm.conf/activation_mode. See **lvmraid(7)** for more information.

--addtag Tag

Adds a tag to a PV, VG or LV. This option can be repeated to add multiple tags at once. See **lvm(8)**

for information about tags.

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--deltag Tag

Deletes a tag from a PV, VG or LV. This option can be repeated to delete multiple tags at once. See **lvm(8)** for information about tags.

--detachprofile

Detaches a metadata profile from a VG or LV. See **lvm.conf(5)** for more information about profiles.

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--ignoreactivationskip

Ignore the "activation skip" LV flag during activation to allow LVs with the flag set to be activated.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--ignoremonitoring

Do not interact with dmeventd unless --monitor is specified. Do not use this if dmeventd is already monitoring a device.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--lockstart

Start the lockspace of a shared VG in lvmlockd. lvmlockd locks becomes available for the VG, allowing LVM to use the VG. See **lvmlockd(8)** for more information.

--lockstop

Stop the lockspace of a shared VG in lvmlockd. lvmlockd locks become unavailable for the VG, preventing LVM from using the VG. See **lvmlockd(8)** for more information.

--locktype sanlock|dlm|none

Change the VG lock type to or from a shared lock type used with lvmlockd. See **lvmlockd(8)** for more information.

-l|--logicalvolume Number

Sets the maximum number of LVs allowed in a VG.

--longhelp

Display long help text.

-p|--maxphysicalvolumes Number

Sets the maximum number of PVs that can belong to the VG. The value 0 removes any limitation. For large numbers of PVs, also see options **--pvmetadatacopies**, and **--vgmetadatacopies** for improving performance.

--metadataprofile String

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--monitor y|n

Start (yes) or stop (no) monitoring an LV with dmeventd. dmeventd monitors kernel events for an LV, and performs automated maintenance for the LV in response to specific events. See **dmeventd(8)** for more information.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

-P|--partial

Commands will do their best to activate LVs with missing PV extents. Missing extents may be replaced with error or zero segments according to the lvm.conf `missing_stripe_filler` setting. Metadata may not be changed with this option.

-s|--physicalextentsize Size[m|UNIT]

Sets the physical extent size of PVs in the VG. The value must be either a power of 2 of at least 1 sector (where the sector size is the largest sector size of the PVs currently used in the VG), or at least 128KiB. Once this value has been set, it is difficult to change without recreating the VG, unless no extents need moving. Before increasing the physical extent size, you might need to use `lvresize`, `pvresize` and/or `pvmove` so that everything fits. For example, every contiguous range of extents used in a LV must start and end on an extent boundary.

--poll y|n

When yes, start the background transformation of an LV. An incomplete transformation, e.g. `pvmove` or `lvconvert` interrupted by reboot or crash, can be restarted from the last checkpoint with **--poll y**. When no, background transformation of an LV will not occur, and the transformation will not complete. It may not be appropriate to immediately poll an LV after activation, in which case **--poll n** can be used to defer polling until a later **--poll y** command.

--profile String

An alias for **--commandprofile** or **--metadataprofile**, depending on the command.

--pvmetadatacopies 0|1|2

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1,

one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--refresh

If the LV is active, reload its metadata. This is not necessary in normal operation, but may be useful if something has gone wrong, or if some form of manual LV sharing is being used.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-x|--resizable y|n

Enables or disables the addition or removal of PVs to/from a VG (by vgextend/vgreduce).

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--sysinit

Indicates that vgchange/lvchange is being invoked from early system initialisation scripts (e.g. rc.sysinit or an initrd), before writable filesystems are available. As such, some functionality needs to be disabled and this option acts as a shortcut which selects an appropriate set of options. Currently, this is equivalent to using --ignorelockingfailure, --ignoremonitoring, --poll n, and setting env var LVM_SUPPRESS_LOCKING_FAILURE_MESSAGES. vgchange/lvchange skip autoactivation, and defer to pvscan autoactivation.

--systemid String

Changes the system ID of the VG. Using this option requires caution because the VG may become foreign to the host running the command, leaving the host unable to access it. See **lvmsystemid(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-u|--uuid

Generate new random UUID for specified VGs.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--[vg]metadatacopies all|unmanaged|Number

Number of copies of the VG metadata that are kept. VG metadata is kept in VG metadata areas on PVs in the VG, i.e. reserved space at the start and/or end of the PVs. Keeping a copy of the VG metadata on every PV can reduce performance in VGs containing a large number of PVs. When this number is set to a non-zero value, LVM will automatically choose PVs on which to store metadata, using the metadataignore flags on PVs to achieve the specified number. The number can also be replaced with special string values: **unmanaged** causes LVM to not automatically manage the PV metadataignore flags. **all** causes LVM to first clear the metadataignore flags on all PVs, and then to become unmanaged.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, **LVM_VG_NAME** can generally be substituted for a required VG parameter.

NOTES

If vgchange recognizes COW snapshot LVs that were dropped because they ran out of space, it displays a message informing the administrator that the snapshots should be removed.

EXAMPLES

Activate all LVs in all VGs on all existing devices.

vgchange -a y

Change the maximum number of LVs for an inactive VG.

vgchange -l 128 vg00

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8)

vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)
lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)
lvm-fullreport(8) lvm-lvpoll(8) lvm2–activation–generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgck – Check the consistency of volume group(s)

SYNOPSIS

```
vgck option_args position_args
      [ option_args ]
      [ position_args ]
```

DESCRIPTION

vgck checks LVM metadata for consistency.

USAGE

Read and display information about a VG.

```
vgck
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
      [ VG|Tag ... ]
```

Rewrite VG metadata to correct problems.

```
vgck --updatemetadata VG
      [ COMMON_OPTIONS ]
```

Common options for command:

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded **y|n**

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--updatemetadata

Update VG metadata to correct problems. If VG metadata was updated while a PV was missing, and the PV reappears with an old version of metadata, then this option (or any other command that writes metadata) will update the metadata on the previously missing PV. If a PV was removed from a VG while it was missing, and the PV reappears, using this option will clear the outdated metadata from the previously missing PV. If metadata text is damaged on one PV, using this option will replace the damaged metadata text. For more severe damage, e.g. with headers, see **pvck(8)**.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is

specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgconvert – Change volume group metadata format

SYNOPSIS

```
vgconvert position_args
      [ option_args ]
```

DESCRIPTION

vgconvert is no longer a part of LVM. It was removed along with support for the LVM1 format. Use an older version of LVM to convert VGs from the LVM1 format to LVM2.

USAGE

```
vgconvert VG ...
      [ -f|--force ]
      [ -M|--metadatatype lvm2 ]
      [ --labelsector Number ]
      [ --bootloaderaresize Size[m|UNIT] ]
      [ --pvmetadatacopies 0|1|2 ]
      [ --metadatasize Size[m|UNIT] ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--bootloaderaresize** *Size[m|UNIT]*

Reserve space for the bootloader between the LVM metadata area and the first PE. The bootloader area is reserved for bootloaders to embed their own data or metadata; LVM will not use it. The bootloader area begins where the first PE would otherwise be located. The first PE is moved out by the size of the bootloader area, and then moved out further if necessary to match the data alignment. The start of the bootloader area is always aligned, see also **--dataalignment** and **--dataalignmentoffset**. The bootloader area may be larger than requested due to the alignment, but it's never less than the requested size. To see the bootloader area start and size of an existing PV use **pvs -o +pv_ba_start,pv_ba_size**.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--labelsector *Number*

By default the PV is labelled with an LVM2 identifier in its second sector (sector 1). This lets you use a different sector near the start of the disk (between 0 and 3 inclusive – see LABEL_SCAN_SECTORS in the source). Use with care.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--metadatasize *Size[m|UNIT]*

The approximate amount of space used for each VG metadata area. The size may be rounded.

-M|--metadatatype lvm2

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format. **lvm1** (or just **1**) is no longer used.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

--pvmetadatacopies 0|1|2

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1, one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgecfgbackup(8) vgecfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgcreate – Create a volume group

SYNOPSIS

```
vgcreate position_args
      [ option_args ]
```

DESCRIPTION

vgcreate creates a new VG on block devices. If the devices were not previously initialized as PVs with **pvcreate(8)**, **vgcreate** will initialize them, making them PVs. The **pvcreate** options for initializing devices are also available with **vgcreate**.

When **vgcreate** uses an existing PV, that PV's existing values for metadata size, PE start, etc, are used, even if different values are specified in the **vgcreate** command. To change these values, first use **pvremove** on the device.

USAGE

```
vgcreate VG_new PV ...
      [ -A|--autobackup y|n ]
      [ -c|--clustered y|n ]
      [ -l|--maxlogicalvolumes Number ]
      [ -p|--maxphysicalvolumes Number ]
      [ -M|--metadatatype lvm2 ]
      [ -s|--physicalextentsize Size[m|UNIT] ]
      [ -f|--force ]
      [ -Z|--zero y|n ]
      [ --addtag Tag ]
      [ --alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit ]
      [ --metadataprofile String ]
      [ --labelsector Number ]
      [ --metadatasize Size[m|UNIT] ]
      [ --pvmetadatacopies 0|1|2 ]
      [ --[vg]metadatacopies all|unmanaged|Number ]
      [ --reportformat basic|json ]
      [ --dataalignment Size[k|UNIT] ]
      [ --dataalignmentoffset Size[k|UNIT] ]
      [ --shared ]
      [ --systemid String ]
      [ --locktype sanlock|dlm|none ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
```

[--version]

OPTIONS

--addtag *Tag*

Adds a tag to a PV, VG or LV. This option can be repeated to add multiple tags at once. See **lvm(8)** for information about tags.

--alloc contiguous|cling|cling_by_tags|normal|anywhere|inherit

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange, or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

-c|--clustered y|n

This option was specific to clvm and is now replaced by the --shared option with **lvmlockd(8)**.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--dataalignment *Size[k|UNIT]*

Align the start of a PV data area with a multiple of this number. To see the location of the first Physical Extent (PE) of an existing PV, use pvs -o +pe_start. In addition, it may be shifted by an alignment offset, see --dataalignmentoffset. Also specify an appropriate PE size when creating a VG.

--dataalignmentoffset *Size[k|UNIT]*

Shift the start of the PV data area by this additional offset.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--labelsector *Number*

By default the PV is labelled with an LVM2 identifier in its second sector (sector 1). This lets you use a different sector near the start of the disk (between 0 and 3 inclusive – see LA-BEL_SCAN_SECTORS in the source). Use with care.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--locktype **sanlock|dlm|none**

Specify the VG lock type directly in place of using --shared. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-l|--maxlogicalvolumes *Number*

Sets the maximum number of LVs allowed in a VG.

-p|--maxphysicalvolumes *Number*

Sets the maximum number of PVs that can belong to the VG. The value 0 removes any limitation. For large numbers of PVs, also see options --pvmetadacopies, and --vgmetadacopies for improving performance.

--metadataprofile *String*

The metadata profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--metadatasize *Size[m|UNIT]*

The approximate amount of space used for each VG metadata area. The size may be rounded.

-M|--metadatatype **lvm2**

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format. **lvm1** (or just **1**) is no longer used.

--nolocking

Disable locking.

-s|--physicalextentsize *Size[m|UNIT]*

Sets the physical extent size of PVs in the VG. The value must be either a power of 2 of at least 1 sector (where the sector size is the largest sector size of the PVs currently used in the VG), or at least 128KiB. Once this value has been set, it is difficult to change without recreating the VG, unless no extents need moving.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

--pvmetadacopies **0|1|2**

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1, one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--shared

Create a shared VG using lvmlockd if LVM is compiled with lockd support. lvmlockd will select lock type sanlock or dlm depending on which lock manager is running. This allows multiple hosts to share a VG on shared devices. lvmlockd and a lock manager must be configured and running. See **lvmlockd(8)** for more information about shared VGs.

--systemid *String*

Specifies the system ID that will be given to the new VG, overriding the system ID of the host

running the command. A VG is normally created without this option, in which case the new VG is given the system ID of the host creating it. Using this option requires caution because the system ID of the new VG may not match the system ID of the host running the command, leaving the VG inaccessible to the host. See **lvmsystemid(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--[vg]metadatacopies all|unmanaged|Number

Number of copies of the VG metadata that are kept. VG metadata is kept in VG metadata areas on PVs in the VG, i.e. reserved space at the start and/or end of the PVs. Keeping a copy of the VG metadata on every PV can reduce performance in VGs containing a large number of PVs. When this number is set to a non-zero value, LVM will automatically choose PVs on which to store metadata, using the metadataignore flags on PVs to achieve the specified number. The number can also be replaced with special string values: **unmanaged** causes LVM to not automatically manage the PV metadataignore flags. **all** causes LVM to first clear the metadataignore flags on all PVs, and then to become unmanaged.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

-Z|--zero y|n

Controls if the first 4 sectors (2048 bytes) of the device are wiped. The default is to wipe these sectors unless either or both of **--restorefile** or **--uuid** are specified.

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkKmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Create a VG with two PVs, using the default physical extent size.
vgcreate myvg /dev/sdk1 /dev/sdl1

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgdisplay – Display volume group information

SYNOPSIS

```
vgdisplay
[ option_args ]
[ position_args ]
```

DESCRIPTION

vgdisplay shows the attributes of VGs, and the associated PVs and LVs.

vgs(8) is a preferred alternative that shows the same information and more, using a more compact and configurable output format.

USAGE

```
vgdisplay
[ -A|--activevolumegroups ]
[ -c|--colon ]
[ -C|--columns ]
[ -o|--options String ]
[ -S|--select String ]
[ -s|--short ]
[ -O|--sort String ]
[ --aligned ]
[ --binary ]
[ --configreport log|vg|lv|pv|pvseg|seg ]
[ --foreign ]
[ --ignoreclockingfailure ]
[ --logonly ]
[ --noheadings ]
[ --nosuffix ]
[ --readonly ]
[ --reportformat basic|json ]
[ --shared ]
[ --separator String ]
[ --unbuffered ]
[ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
[ COMMON_OPTIONS ]
[ VG|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**-A|--activevolumegroups**

Only select active VGs. The VG is considered active if at least one of its LVs is active.

--aligned

Use with --separator to align the output columns

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

-c|--colon

Generate colon separated output for easier parsing in scripts or programs. Also see **vgs(8)** which provides considerably more control over the output.

-C|--columns

Display output in columns, the equivalent of **vgs(8)**. Options listed are the same as options given in **vgs(8)**.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport *log|vg|lv|pv|pvseg|seg*

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with **--units** (except h and H) if processing the output.

-o|--options String

Comma-separated, ordered list of fields to display in columns. String arg syntax is:
`[+|-#]Field1[,Field2 ...]` The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the **-o** option: `-o+field1,field2 -o-field3,field4 -o#field5`. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **Ivmreport(7)** for more information about reporting.

--profile String

An alias for **--commandprofile** or **--metadataprofile**, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides **--debug** and **--verbose**. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **Ivmreport(7)** for more information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by **--select help** and **Ivmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See **--options help** for selectable object fields. Rows can be displayed with an additional "selected" field (**-o selected**) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator String

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when lvmlockd is not being used on the host. See **Ivmlockd(8)** for more information about shared VGs.

-s|--short

Give a short listing showing the existence of VGs.

-O|--sort String

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any column with - for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-readable, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes. Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g. --units 3M.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgexport – Unregister volume group(s) from the system

SYNOPSIS

```
vgexport option_args position_args
      [ option_args ]
```

DESCRIPTION

vgexport changes a VG into the exported state, which ensures that the VG and its disks are not being used, and cannot be used until the VG is imported by **vgimport**(8). Putting a VG into an unusable, offline state can be useful when doing things like moving a VG's disks to another system. Exporting a VG provides some protection from its LVs being accidentally used, or being used by an automated system before it's ready.

A VG cannot be exported until all of its LVs are inactive.

LVM commands will ignore an exported VG or report an error if a command tries to use it.

For an exported VG, the **vgs** command will display attribute, and the **pvs** command will display attribute. Both **vgs** and **pvs** will display report field.

vgexport clears the VG system ID, and **vgimport** sets the VG system ID to match the host running **vgimport** (if the host has a system ID).

USAGE

Export specified VGs.

```
vgexport VG|Tag|Select ...
      [ -S|--select String ]
      [ COMMON_OPTIONS ]
```

Export all VGs.

```
vgexport -a|--all
      [ COMMON_OPTIONS ]
```

Common options for command:

```
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-a|--all

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadaprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat *basic|json*

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select *String*

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **--qq**.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgextend – Add physical volumes to a volume group

SYNOPSIS

```
vgextend position_args
      [ option_args ]
```

DESCRIPTION

vgextend adds one or more PVs to a VG. This increases the space available for LVs in the VG.

Also, PVs that have gone missing and then returned, e.g. due to a transient device failure, can be added back to the VG without re-initializing them (see **--restoremissing**).

If the specified PVs have not yet been initialized with **pvcreate**, **vgextend** will initialize them. In this case **pvcreate** options can be used, e.g. **--labelsector**, **--metadatasize**, **--metadataignore**, **--pvmetadatacopies**, **--dataalignment**, **--dataalignmentoffset**.

USAGE

```
vgextend VG PV ...
      [ -A|--autobackup y|n ]
      [ -f|--force ]
      [ -Z|--zero y|n ]
      [ -M|--metadatatype lvm2 ]
      [ --labelsector Number ]
      [ --metadatasize Size[m|UNIT] ]
      [ --pvmetadatacopies 0|1|2 ]
      [ --metadataignore y|n ]
      [ --dataalignment Size[k|UNIT] ]
      [ --dataalignmentoffset Size[k|UNIT] ]
      [ --reportformat basic|json ]
      [ --restoremissing ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--dataalignment *Size[k|UNIT]*

Align the start of a PV data area with a multiple of this number. To see the location of the first Physical Extent (PE) of an existing PV, use pvs -o +pe_start. In addition, it may be shifted by an alignment offset, see --dataalignmentoffset. Also specify an appropriate PE size when creating a VG.

--dataalignmentoffset *Size[k|UNIT]*

Shift the start of the PV data area by this additional offset.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--labelsector *Number*

By default the PV is labelled with an LVM2 identifier in its second sector (sector 1). This lets you use a different sector near the start of the disk (between 0 and 3 inclusive – see LA-BEL_SCAN_SECTORS in the source). Use with care.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--metadataignore *y|n*

Specifies the metadataignore property of a PV. If yes, metadata areas on the PV are ignored, and lvm will not store metadata in the metadata areas of the PV. If no, lvm will store metadata on the PV.

--metadatasize *Size[m|UNIT]*

The approximate amount of space used for each VG metadata area. The size may be rounded.

-M|--metadatatype *lvm2*

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format. **lvm1** (or just **1**) is no longer used.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

--pvmetadatacopies *0|1|2*

The number of metadata areas to set aside on a PV for storing VG metadata. When 2, one copy of the VG metadata is stored at the front of the PV and a second copy is stored at the end. When 1, one copy of the VG metadata is stored at the front of the PV. When 0, no copies of the VG metadata are stored on the given PV. This may be useful in VGs containing many PVs (this places limitations on the ability to use vgsplit later.)

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--restoremissing

Add a PV back into a VG after the PV was missing and then returned, e.g. due to a transient failure. The PV is not reinitialized.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

-Z|--zero y|n

Controls if the first 4 sectors (2048 bytes) of the device are wiped. The default is to wipe these sectors unless either or both of --restorefile or --uuid are specified.

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Add two PVs to a VG.

```
vgextend vg00 /dev/sda4 /dev/sdn1
```

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgimport – Register exported volume group with system

SYNOPSIS

```
vgimport option_args position_args  
[ option_args ]
```

DESCRIPTION

vgimport makes exported VGs known to the system again, perhaps after moving the PVs from a different system.

vgexport clears the VG system ID, and vgimport sets the VG system ID to match the host running vgimport (if the host has a system ID).

USAGE

Import specified VGs.

```
vgimport VG|Tag|Select ...  
[ -S|--select String ]  
[ COMMON_OPTIONS ]
```

Import all VGs.

```
vgimport -a|--all  
[ COMMON_OPTIONS ]
```

Common options for command:

```
[ -f|--force ]  
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]  
[ -h|--help ]  
[ -q|--quiet ]  
[ -t|--test ]  
[ -v|--verbose ]  
[ -y|--yes ]  
[ --commandprofile String ]  
[ --config String ]  
[ --driverloaded y|n ]  
[ --lockopt String ]  
[ --longhelp ]  
[ --nolocking ]  
[ --profile String ]  
[ --version ]
```

OPTIONS

-a|--all

Import all visible VGs.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES**VG**

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used. No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisk(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

`vgimportclone` – Import a VG from cloned PVs

SYNOPSIS

```
vgimportclone position_args
    [ option_args ]
```

DESCRIPTION

`vgimportclone` imports a VG from duplicated PVs, e.g. created by a hardware snapshot of existing PVs.

A duplicated VG cannot be used until it is made to coexist with the original VG. `vgimportclone` renames the VG associated with the specified PVs and changes the associated VG and PV UUIDs.

USAGE

```
vgimportclone PV ...
    [ -n|--basevgname VG ]
    [ -i|--import ]
    [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-n|--basevgname *String*

By default the snapshot VG will be renamed to the original name plus a numeric suffix to avoid duplicate naming (e.g. 'test_vg' would be renamed to 'test_vg1'). This option will override the base VG name that is used for all VG renames. If a VG already exists with the specified name a numeric suffix will be added (like the previous example) to make it unique.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

-i|--import

Import exported VGs. Otherwise VGs that have been exported will not be changed (nor will their associated PVs).

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*PV*

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

An original VG "vg00" has PVs "/dev/sda" and "/dev/sdb". The corresponding PVs from a hardware snapshot are "/dev/sdc" and "/dev/sdd". Rename the VG associated with "/dev/sdc" and "/dev/sdd" from "vg00"

to "vg00_snap" (and change associated UUIDs).

vgimportclone --basevgname vg00_snap /dev/sdc /dev/sdd

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisk(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgmerge – Merge volume groups

SYNOPSIS

```
vgmerge position_args
      [ option_args ]
```

DESCRIPTION

vgmerge merges two existing VGs. The inactive source VG is merged into the destination VG if physical extent sizes are equal and PV and LV summaries of both VGs fit into the destination VG's limits.

USAGE

```
vgmerge VG VG
      [ -A|--autobackup y|n ]
      [ -l|--list ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

-l|--list

Display merged destination VG like vgdisplay -v.

--lockopt *String*
 Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp
 Display long help text.

--nolocking
 Disable locking.

--profile *String*
 An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...
 Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

-t|--test
 Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...
 Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version
 Display version information.

-y|--yes
 Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Merge an inactive VG named "vg00" into the active or inactive VG named "databases", giving verbose run-time information.

vgmerge -v databases vg00

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8)

vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgmknodes – Create the special files for volume group devices in /dev

SYNOPSIS

```
vgmknodes
[ option_args ]
[ position_args ]
```

DESCRIPTION

vgmknodes checks the LVM device nodes in /dev that are needed for active LVs and creates any that are missing and removes unused ones.

This command should not usually be needed if all the system components are interoperating correctly.

USAGE

```
vgmknodes
[ --ignorelockingfailure ]
[ --refresh ]
[ --reportformat basic|json ]
[ COMMON_OPTIONS ]
[ VG|LV|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded **y|n**

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*
 Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp
 Display long help text.

--nolocking
 Disable locking.

--profile *String*
 An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...
 Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--refresh
 If the LV is active, reload its metadata. This is not necessary in normal operation, but may be useful if something has gone wrong, or if some form of manual LV sharing is being used.

--reportformat basic|json
 Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test
 Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...
 Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version
 Display version information.

-y|--yes
 Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see --qq.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

LV

Logical Volume name. See **lvm(8)** for valid names. An LV positional arg generally includes the VG name and LV name, e.g. VG/LV.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where

capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm**(8) for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) **lvm.conf**(5) **lvmconfig**(8)

pvchange(8) **pvck**(8) **pvcreate**(8) **pvdisplay**(8) **pvmove**(8) **pvremove**(8) **pvresize**(8) **pvs**(8) **pvscan**(8)

vgecfgbackup(8) **vgecfgrestore**(8) **vgchange**(8) **vgck**(8) **vgcreate**(8) **vgconvert**(8) **vgdisplay**(8) **vgexport**(8) **vgextend**(8) **vgimport**(8) **vgimportclone**(8) **vgmerge**(8) **vgmknodes**(8) **vgreduce**(8) **vgremove**(8) **vgrename**(8) **vgs**(8) **vgscan**(8) **vgsplit**(8)

lvcreate(8) **lvchange**(8) **lvconvert**(8) **lvdisplay**(8) **lvextend**(8) **lvreduce**(8) **lvremove**(8) **lvrename**(8) **lvresize**(8) **lvs**(8) **lvscan**(8)

lvm-fullreport(8) **lvm-lvpoll**(8) **lvm2-activation-generator**(8) **blkdeactivate**(8) **lvmdump**(8)

dmeventd(8) **lvmpolld**(8) **lvmlockd**(8) **lvmlockctl**(8) **cmirrord**(8) **lvmdbusd**(8)

lvmsystemid(7) **lvmreport**(7) **lvmraid**(7) **lvmthin**(7) **lvmcache**(7)

NAME

vgreduce – Remove physical volume(s) from a volume group

SYNOPSIS

```
vgreduce option_args position_args
      [ option_args ]
      -a|--all
      -A|--autobackup y|n
      --commandprofile String
      --config String
      -d|--debug
      --driverloaded y|n
      -f|--force
      -h|--help
      --lockopt String
      --longhelp
      --mirrorsonly
      --nolocking
      --profile String
      -q|--quiet
      --removemissing
      --reportformat basic|json
      -t|--test
      -v|--verbose
      --version
      -y|--yes
```

DESCRIPTION

vgreduce removes one or more unused PVs from a VG.

USAGE

Remove a PV from a VG.

```
vgreduce VG PV ...
      [ COMMON_OPTIONS ]
```

Remove all unused PVs from a VG.

```
vgreduce -a|--all VG
      [ COMMON_OPTIONS ]
```

Remove all missing PVs from a VG.

```
vgreduce --removemissing VG
      [ --mirrorsonly ]
      [ COMMON_OPTIONS ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
```

```
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**-a|--all**

Removes all empty PVs if none are named on the command line.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--mirrorsonly

Only remove missing PVs from mirror LVs.

--nolocking

Disable locking.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--removemissing

Removes all missing PVs from the VG, if there are no LVs allocated on them. This resumes normal operation of the VG (new LVs may again be created, changed and so on). If this is not possible because LVs are referencing the missing PVs, this option can be combined with --force to have the command remove any partial LVs. In this case, any LVs and dependent snapshots that were partly on the missing disks are removed completely, including those parts on disks that are still present. If LVs spanned several disks, including ones that are lost, salvaging some data first may be possible by activating LVs in partial mode.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see --qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) **lvm.conf(5)** **lvmconfig(8)**

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)
vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)
lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)
lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgremove – Remove volume group(s)

SYNOPSIS

```
vgremove position_args
      [ option_args ]
```

DESCRIPTION

vgremove removes one or more VGs. If LVs exist in the VG, a prompt is used to confirm LV removal.

If one or more PVs in the VG are lost, consider **vgreduce --removemissing** to make the VG metadata consistent again.

Repeat the force option (**-ff**) to forcibly remove LVs in the VG without confirmation.

USAGE

```
vgremove VG|Tag|Select ...
      [ -f|--force ]
      [ -S|--select String ]
      [ --noudevsync ]
      [ --reportformat basic|json ]
      [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded **y|n**

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--noudevsync

Disables udev synchronisation. The process will not wait for notification from udev. It will continue irrespective of any possible udev processing in the background. Only use this if udev is not running or has rules that ignore the devices LVM creates.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-S|--select String

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

Select

Select indicates that a required positional parameter can be omitted if the **--select** option is used.
No arg appears in this position.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgrename – Rename a volume group

SYNOPSIS

```
vgrename position_args
      [ option_args ]
```

DESCRIPTION

vgrename renames a VG.

All VGs visible to a system need to have different names, otherwise many LVM commands will refuse to run or give warning messages. VGs with the same name can occur when disks are moved between machines, or filters are changed. If a newly connected disk has a VG with the same name as the VG containing the root filesystem, the machine may not boot correctly. When two VGs have the same name, the VG UUID can be used in place of the source VG name.

USAGE

Rename a VG.

```
vgrename VG VG_new
      [ COMMON_OPTIONS ]
```

Rename a VG by specifying the VG UUID.

```
vgrename String VG_new
      [ COMMON_OPTIONS ]
```

Common options for command:

```
[ -A|--autobackup y|n ]
[ -f|--force ]
[ --reportformat basic|json ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS

-A|--autobackup *y|n*

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgcfgbackup(8)** for more information.

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same

format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-f|--force ...

Override various checks, confirmations and protections. Use with extreme caution.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--nolocking

Disable locking.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES

VG

Volume Group name. See **lvm(8)** for valid names.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is

specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

EXAMPLES

Rename VG "vg02" to "myvg":

```
vgrename vg02 myvg
```

Rename the VG with the specified UUID to "myvg".

```
vgrename Zvlifi-Ep3t-e0Ng-U42h-o0ye-KHu1-nl7Ns4 myvg
```

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgecfgbackup(8) vgecfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)

lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgs – Display information about volume groups

SYNOPSIS

```
vgs
  [ option_args ]
  [ position_args ]
```

DESCRIPTION

vgs produces formatted output about VGs.

USAGE

```
vgs
  [ -a|--all ]
  [ -o|--options String ]
  [ -S|--select String ]
  [ -O|--sort String ]
  [ --aligned ]
  [ --binary ]
  [ --configreport log|vg|lv|pv|pvseg|seg ]
  [ --foreign ]
  [ --ignorelockingfailure ]
  [ --logonly ]
  [ --nameprefixes ]
  [ --noheadings ]
  [ --nosuffix ]
  [ --readonly ]
  [ --reportformat basic|json ]
  [ --rows ]
  [ --separator String ]
  [ --shared ]
  [ --unbuffered ]
  [ --units r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E ]
  [ --unquoted ]
  [ COMMON_OPTIONS ]
  [ VG|Tag ... ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--aligned**

Use with --separator to align the output columns

-a|--all

List all VGs. Equivalent to not specifying any VGs.

--binary

Use binary values "0" or "1" instead of descriptive literal values for columns that have exactly two valid values to report (not counting the "unknown" value which denotes that the value could not be determined).

--commandprofile *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

--configreport log|vg|lv|pv|pvseg|seg

See **lvmreport(7)**.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

--foreign

Report/display foreign VGs that would otherwise be skipped. See **lvmsystemid(7)** for more information about foreign VGs.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--logonly

Suppress command report and display only log report.

--longhelp

Display long help text.

--nameprefixes

Add an "LVM2_" prefix plus the field name to the output. Useful with --noheadings to produce a list of field=value pairs that can be used to set environment variables (for example, in udev rules).

--noheadings

Suppress the headings line that is normally the first line of output. Useful if grepping the output.

--nolocking

Disable locking.

--nosuffix

Suppress the suffix on output sizes. Use with --units (except h and H) if processing the output.

-o|--options *String*

Comma-separated, ordered list of fields to display in columns. String arg syntax is:
[+|-#]Field1[,Field2 ...] The prefix + will append the specified fields to the default fields, - will remove the specified fields from the default fields, and # will compact specified fields (removing them when empty for all rows.) Use **-o help** to view the list of all available fields. Use separate lists of fields to add, remove or compact by repeating the -o option: -o+field1,field2 -o-

field3,field4 -o#field5. These lists are evaluated from left to right. Use field name **lv_all** to view all LV fields, **vg_all** all VG fields, **pv_all** all PV fields, **pvseg_all** all PV segment fields, **seg_all** all LV segment fields, and **pvseg_all** all PV segment columns. See the lvm.conf report section for more config options. See **lvmreport(7)** for more information about reporting.

--profile *String*

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--readonly

Run the command in a special read-only mode which will read on-disk metadata without needing to take any locks. This can be used to peek inside metadata used by a virtual machine image while the virtual machine is running. No attempt will be made to communicate with the device-mapper kernel driver, so this option is unable to report whether or not LVs are actually in use.

--reportformat **basic|json**

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

--rows

Output columns as rows.

-S|--select *String*

Select objects for processing and reporting based on specified criteria. The criteria syntax is described by --select help and **lvmreport(7)**. For reporting commands, one row is displayed for each object matching the criteria. See --options help for selectable object fields. Rows can be displayed with an additional "selected" field (-o selected) showing 1 if the row matches the selection and 0 otherwise. For non-reporting commands which process LVM entities, the selection is used to choose items to process.

--separator *String*

String to use to separate each column. Useful if grepping the output.

--shared

Report/display shared VGs that would otherwise be skipped when lvmlockd is not being used on the host. See **lvmlockd(8)** for more information about shared VGs.

-O|--sort *String*

Comma-separated ordered list of columns to sort by. Replaces the default selection. Precede any column with - for a reverse sort on that column.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

--unbuffered

Produce output immediately without sorting or aligning the columns properly.

--units **r|R|h|H|b|B|s|S|k|K|m|M|g|G|t|T|p|P|e|E**

All sizes are output in these units: human-(r)eadable with '<' rounding indicator, (h)uman-readable, (b)ytes, (s)ectors, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (p)etabytes, (e)xabytes. Capitalise to use multiples of 1000 (S.I.) instead of 1024. Custom units can be specified, e.g.

--units 3M.

--unquoted

When used with --nameprefixes, output values in the field=value pairs are not quoted.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

Tag

Tag name. See **lvm(8)** for information about tag names and using tags in place of a VG, LV or PV.

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

NOTES

The vg_attr bits are:

- 1 Permissions: (w)ritable, (r)ead-only
- 2 Resi(z)eable
- 3 E(x)ported
- 4 (p)artial: one or more physical volumes belonging to the volume group are missing from the system
- 5 Allocation policy: (c)ontiguous, c(l)ing, (n)ormal, (a)nywhere
- 6 (c)lustered, (s)hared

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)

lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmddump(8)

dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgscan – Search for all volume groups

SYNOPSIS

```
vgscan
  [ option_args ]
```

DESCRIPTION

vgscan scans all supported LVM block devices in the system for VGs.

USAGE

```
vgscan
  [ --ignorelockingfailure ]
  [ --mknodes ]
  [ --notifydbus ]
  [ --reportformat basic|json ]
  [ COMMON_OPTIONS ]
```

Common options for lvm:

```
[ -d|--debug ]
[ -h|--help ]
[ -q|--quiet ]
[ -t|--test ]
[ -v|--verbose ]
[ -y|--yes ]
[ --commandprofile String ]
[ --config String ]
[ --driverloaded y|n ]
[ --lockopt String ]
[ --longhelp ]
[ --nolocking ]
[ --profile String ]
[ --version ]
```

OPTIONS**--commandprofile** *String*

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config *String*

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded *y|n*

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--ignorelockingfailure

Allows a command to continue with read-only metadata operations after locking failures.

--lockopt *String*

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

--mknodes

Also checks the LVM special files in /dev that are needed for active LVs and creates any missing ones and removes unused ones.

--nolocking

Disable locking.

--notifydbus

Send a notification to D-Bus. The command will exit with an error if LVM is not built with support for D-Bus notification, or if the notify_dbus config setting is disabled.

--profile String

An alias for --commandprofile or --metadataprofile, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides --debug and --verbose. Repeat once to also suppress any prompts with answer 'no'.

--reportformat basic|json

Overrides current output format for reports which is defined globally by the report/output_format setting in lvm.conf. **basic** is the original format with columns and rows. If there is more than one report per command, each report is prefixed with the report name for identification. **json** produces report output in JSON format. See **lvmreport(7)** for more information.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see -qq.)

VARIABLES*String*

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmGgTtPpEe**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control --units, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, LVM_VG_NAME can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)
vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)
lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8) lvresize(8) lvs(8) lvscan(8)
lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vgsplit – Move physical volumes into a new or existing volume group

SYNOPSIS

```
vgsplit option_args position_args
      [ option_args ]
```

DESCRIPTION

vgsplit moves one or more PVs from a source VG (the first VG arg) to a destination VG (the second VG arg). The PV(s) to move are named after the source and destination VGs, or an LV is named, in which case the PVs underlying the LV are moved.

If the destination VG does not exist, a new VG is created (command options can be used to specify properties of the new VG, also see **vgcreate(8)**.)

LVs cannot be split between VGs; each LV must be entirely on the PVs in the source or destination VG.

vgsplit can only move complete PVs. (See **pvmove(8)** for moving part of a PV.)

USAGE

Split a VG by specified PVs.

```
vgsplit VG VG PV ...
      [ COMMON_OPTIONS ]
```

Split a VG by PVs in a specified LV.

```
vgsplit -n|--name LV VG VG
      [ COMMON_OPTIONS ]
```

Common options for command:

- [**-A|--autobackup** *y|n*]
- [**-l|--maxlogicalvolumes** *Number*]
- [**-p|--maxphysicalvolumes** *Number*]
- [**-M|--metadatatype** *lvm2*]
- [**--alloc** *contiguous|cling|cling_by_tags|normal|anywhere|inherit*]
- [**--[vg]metadatacopies** *all|unmanaged|Number*]

Common options for lvm:

- [**-d|--debug**]
- [**-h|--help**]
- [**-q|--quiet**]
- [**-t|--test**]
- [**-v|--verbose**]
- [**-y|--yes**]
- [**--commandprofile** *String*]
- [**--config** *String*]
- [**--driverloaded** *y|n*]
- [**--lockopt** *String*]
- [**--longhelp**]
- [**--nolocking**]
- [**--profile** *String*]
- [**--version**]

OPTIONS

--alloc *contiguous|cling|cling_by_tags|normal|anywhere|inherit*

Determines the allocation policy when a command needs to allocate Physical Extents (PEs) from the VG. Each VG and LV has an allocation policy which can be changed with vgchange/lvchange,

or overridden on the command line. **normal** applies common sense rules such as not placing parallel stripes on the same PV. **inherit** applies the VG policy to an LV. **contiguous** requires new PEs be placed adjacent to existing PEs. **cling** places new PEs on the same PV as existing PEs in the same stripe of the LV. If there are sufficient PEs for an allocation, but normal does not use them, **anywhere** will use them even if it reduces performance, e.g. by placing two stripes on the same PV. Optional positional PV args on the command line can also be used to limit which PVs the command will use for allocation. See **lvm(8)** for more information about allocation.

-A|--autobackup y|n

Specifies if metadata should be backed up automatically after a change. Enabling this is strongly advised! See **vgecfgbackup(8)** for more information.

--commandprofile String

The command profile to use for command configuration. See **lvm.conf(5)** for more information about profiles.

--config String

Config settings for the command. These override lvm.conf settings. The String arg uses the same format as lvm.conf, or may use section/field syntax. See **lvm.conf(5)** for more information about config.

-d|--debug ...

Set debug level. Repeat from 1 to 6 times to increase the detail of messages sent to the log file and/or syslog (if configured).

--driverloaded y|n

If set to no, the command will not attempt to use device-mapper. For testing and debugging.

-h|--help

Display help text.

--lockopt String

Used to pass options for special cases to lvmlockd. See **lvmlockd(8)** for more information.

--longhelp

Display long help text.

-l|--maxlogicalvolumes Number

Sets the maximum number of LVs allowed in a VG.

-p|--maxphysicalvolumes Number

Sets the maximum number of PVs that can belong to the VG. The value 0 removes any limitation. For large numbers of PVs, also see options **--pvmetadacopies**, and **--vgmetadacopies** for improving performance.

-M|--metadatatype lvm2

Specifies the type of on-disk metadata to use. **lvm2** (or just **2**) is the current, standard format.

lvm1 (or just **1**) is no longer used.

-n|--name String

Move only PVs used by the named LV.

--nolocking

Disable locking.

--profile String

An alias for **--commandprofile** or **--metadaprofile**, depending on the command.

-q|--quiet ...

Suppress output and log messages. Overrides **--debug** and **--verbose**. Repeat once to also suppress any prompts with answer 'no'.

-t|--test

Run in test mode. Commands will not update metadata. This is implemented by disabling all

metadata writing but nevertheless returning success to the calling function. This may lead to unusual error messages in multi-stage operations if a tool relies on reading back metadata it believes has changed but hasn't.

-v|--verbose ...

Set verbose level. Repeat from 1 to 4 times to increase the detail of messages sent to stdout and stderr.

--version

Display version information.

--[vg]metadatacopies all|unmanaged|Number

Number of copies of the VG metadata that are kept. VG metadata is kept in VG metadata areas on PVs in the VG, i.e. reserved space at the start and/or end of the PVs. Keeping a copy of the VG metadata on every PV can reduce performance in VGs containing a large number of PVs. When this number is set to a non-zero value, LVM will automatically choose PVs on which to store metadata, using the metadataignore flags on PVs to achieve the specified number. The number can also be replaced with special string values: **unmanaged** causes LVM to not automatically manage the PV metadataignore flags. **all** causes LVM to first clear the metadataignore flags on all PVs, and then to become unmanaged.

-y|--yes

Do not prompt for confirmation interactively but always assume the answer yes. Use with extreme caution. (For automatic no, see **-qq**.)

VARIABLES*VG*

Volume Group name. See **lvm(8)** for valid names.

PV

Physical Volume name, a device path under /dev. For commands managing physical extents, a PV positional arg generally accepts a suffix indicating a range (or multiple ranges) of physical extents (PEs). When the first PE is omitted, it defaults to the start of the device, and when the last PE is omitted it defaults to end. Start and end range (inclusive): *PV[:PE-PE]...* Start and length range (counting from 0): *PV[:PE+PE]...*

String

See the option description for information about the string content.

Size[UNIT]

Size is an input number that accepts an optional unit. Input units are always treated as base two values, regardless of capitalization, e.g. 'k' and 'K' both refer to 1024. The default input unit is specified by letter, followed by |UNIT. UNIT represents other possible input units:**bBsSkMmMg-GtTpPeE**. b|B is bytes, s|S is sectors of 512 bytes, k|K is KiB, m|M is MiB, g|G is GiB, t|T is TiB, p|P is PiB, e|E is EiB. (This should not be confused with the output control **--units**, where capital letters mean multiple of 1000.)

ENVIRONMENT VARIABLES

See **lvm(8)** for information about environment variables used by lvm. For example, **LVM_VG_NAME** can generally be substituted for a required VG parameter.

SEE ALSO

lvm(8) lvm.conf(5) lvmconfig(8)

pvchange(8) pvck(8) pvcreate(8) pvdisplay(8) pvmove(8) pvremove(8) pvresize(8) pvs(8) pvscan(8)

vgcfgbackup(8) vgcfgrestore(8) vgchange(8) vgck(8) vgcreate(8) vgconvert(8) vgdisplay(8) vgexport(8) vgextend(8) vgimport(8) vgimportclone(8) vgmerge(8) vgmknodes(8) vgreduce(8) vgremove(8) vgrename(8) vgs(8) vgscan(8) vgsplit(8)

**lvcreate(8) lvchange(8) lvconvert(8) lvdisplay(8) lvextend(8) lvreduce(8) lvremove(8) lvrename(8)
lvresize(8) lvs(8) lvscan(8)**
**lvm-fullreport(8) lvm-lvpoll(8) lvm2-activation-generator(8) blkdeactivate(8) lvmdump(8)
dmeventd(8) lvmpolld(8) lvmlockd(8) lvmlockctl(8) cmirrord(8) lvmdbusd(8)**
lvmsystemid(7) lvmreport(7) lvmraid(7) lvmthin(7) lvmcache(7)

NAME

vim – Vi IMproved, a programmer's text editor

SYNOPSIS

```
vim [options] [file ..]
vim [options] -
vim [options] -t tag
vim [options] -q [errorfile]

ex
view
gvim gview evim eview
rvim rview rgvim rgview
```

DESCRIPTION

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between **Vim** and Vi.

While running **Vim** a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.

Most often **Vim** is started to edit a single file with the command

```
vim file
```

More generally **Vim** is started with:

```
vim [options] [filelist]
```

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

- | | |
|----------------|---|
| file .. | A list of filenames. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. You can get to the other files with the ":next" command. To edit a file that starts with a dash, precede the filelist with "--". |
| - | The file to edit is read from stdin. Commands are read from stderr, which should be a tty. |
| -t {tag} | The file to edit and the initial cursor position depends on a "tag", a sort of goto label. {tag} is looked up in the tags file, the associated file becomes the current file and the associated command is executed. Mostly this is used for C programs, in which case {tag} could be a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function. See ":help tag-commands". |
| -q [errorfile] | Start in quickFix mode. The file [errorfile] is read and the first error is displayed. If [errorfile] is omitted, the filename is obtained from the 'errorfile' option (defaults to "AztecC.Err" for the Amiga, "errors.err" on other systems). Further errors can be jumped to with the ":cn" command. See ":help quickfix". |

Vim behaves differently, depending on the name of the command (the executable may still be the same file).

- | | |
|------|---|
| vim | The "normal" way, everything is default. |
| ex | Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "-e" argument. |
| view | Start in read-only mode. You will be protected from writing the files. Can also be done with the "-R" argument. |

gvim gview

The GUI version. Starts a new window. Can also be done with the "-g" argument.

evim eview

The GUI version in easy mode. Starts a new window. Can also be done with the "-y" argument.

rvim rview rgvim rgview

Like the above, but with restrictions. It will not be possible to start shell commands, or suspend **Vim**. Can also be done with the "-Z" argument.

OPTIONS

The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash.

- +[num] For the first file the cursor will be positioned on line "num". If "num" is missing, the cursor will be positioned on the last line.
- +/ {pat} For the first file the cursor will be positioned in the line with the first occurrence of {pat}. See ":help search-pattern" for the available search patterns.
- +{command}
- c {command} {command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be enclosed in double quotes (this depends on the shell that is used). Example: vim "+set si" main.c
Note: You can use up to 10 "+" or "-c" commands.
- S {file} {file} will be sourced after the first file has been read. This is equivalent to -c "source {file} ". {file} cannot start with '-'. If {file} is omitted "Session.vim" is used (only works when -S is the last argument).
- cmd {command} Like using "-c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "-c" commands.
- A If **Vim** has been compiled with ARABIC support for editing right-to-left oriented files and Arabic keyboard mapping, this option starts **Vim** in Arabic mode, i.e. 'arabic' is set. Otherwise an error message is given and **Vim** aborts.
- b Binary mode. A few options will be set that makes it possible to edit a binary or executable file.
- C Compatible. Set the 'compatible' option. This will make **Vim** behave mostly like Vi, even though a .vimrc file exists.
- d Start in diff mode. There should between two to eight file name arguments. **Vim** will open all the files and show differences between them. Works like vimdiff(1).
- d {device} Open {device} for use as a terminal. Only on the Amiga. Example: "-d con:20/30/600/150".
- D Debugging. Go to debugging mode when executing the first command from a script.
- e Start **Vim** in Ex mode, just like the executable was called "ex".
- E Start **Vim** in improved Ex mode, just like the executable was called "exim".
- f Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in. On the Amiga, **Vim** is not restarted to open a new window. This option should be used when **Vim** is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and ":!" commands will not work.
- nofork Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in.

- F If **Vim** has been compiled with FKMAP support for editing right-to-left oriented files and Farsi keyboard mapping, this option starts **Vim** in Farsi mode, i.e. 'fkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.
- g If **Vim** has been compiled with GUI support, this option enables the GUI. If no GUI support was compiled in, an error message is given and **Vim** aborts.
- h Give a bit of help about the command line arguments and options. After this **Vim** exits.
- H If **Vim** has been compiled with RIGHTLEFT support for editing right-to-left oriented files and Hebrew keyboard mapping, this option starts **Vim** in Hebrew mode, i.e. 'hkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.
- i {viminfo} Specifies the filename to use when reading or writing the viminfo file, instead of the default "~/viminfo". This can also be used to skip the use of the .viminfo file, by giving the name "NONE".
- L Same as -r.
- l Lisp mode. Sets the 'lisp' and 'showmatch' options on.
- m Modifying files is disabled. Resets the 'write' option. You can still modify the buffer, but writing a file is not possible.
- M Modifications not allowed. The 'modifiable' and 'write' options will be unset, so that changes are not allowed and files can not be written. Note that these options can be set to enable making modifications.
- N No-compatible mode. Resets the 'compatible' option. This will make **Vim** behave a bit better, but less Vi compatible, even though a .vimrc file does not exist.
- n No swap file will be used. Recovery after a crash will be impossible. Handy if you want to edit a file on a very slow medium (e.g. floppy). Can also be done with ":set uc=0". Can be undone with ":set uc=200".
- nb Become an editor server for NetBeans. See the docs for details.
- o[N] Open N windows stacked. When N is omitted, open one window for each file.
- O[N] Open N windows side by side. When N is omitted, open one window for each file.
- p[N] Open N tab pages. When N is omitted, open one tab page for each file.
- R Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from accidentally overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The -R option also implies the -n option (see above). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'".
- r List swap files, with information about using them for recovery.
- r {file} Recovery mode. The swap file is used to recover a crashed editing session. The swap file is a file with the same filename as the text file with ".swp" appended. See ":help recovery".
- s Silent mode. Only when started as "Ex" or when the "-e" option was given before the "-s" option.
- s {scriptin} The script file {scriptin} is read. The characters in the file are interpreted as if you had typed them. The same can be done with the command ":source! {scriptin}". If the end of the file is reached before the editor exits, further characters are read from the keyboard.
- T {terminal} Tells **Vim** the name of the terminal you are using. Only required when the automatic way doesn't work. Should be a terminal known to **Vim** (builtin) or defined in the termcap or terminfo file.
- u {vimrc} Use the commands in the file {vimrc} for initializations. All the other initializations are skipped. Use this to edit a special kind of files. It can also be used to skip all initializations

by giving the name "NONE". See ":help initialization" within vim for more details.

- U {gvimrc} Use the commands in the file {gvimrc} for GUI initializations. All the other GUI initializations are skipped. It can also be used to skip all GUI initializations by giving the name "NONE". See ":help gui-init" within vim for more details.
- V[N] Verbose. Give messages about which files are sourced and for reading and writing a viminfo file. The optional number N is the value for 'verbose'. Default is 10.
- v Start **Vim** in Vi mode, just like the executable was called "vi". This only has effect when the executable is called "ex".
- w {scriptout} All the characters that you type are recorded in the file {scriptout}, until you exit **Vim**. This is useful if you want to create a script file to be used with "vim -s" or ":source!". If the {scriptout} file exists, characters are appended.
- W {scriptout} Like -w, but an existing file is overwritten.
- x Use encryption when writing files. Will prompt for a crypt key.
- X Don't connect to the X server. Shortens startup time in a terminal, but the window title and clipboard will not be used.
- y Start **Vim** in easy mode, just like the executable was called "evim" or "eview". Makes **Vim** behave like a click-and-type editor.
- Z Restricted mode. Works like the executable starts with "r".
- Denotes the end of the options. Arguments after this will be handled as a file name. This can be used to edit a filename that starts with a '-'.
- clean Do not use any personal configuration (vimrc, plugins, etc.). Useful to see if a problem reproduces with a clean Vim setup.
- echo-wid GTK GUI only: Echo the Window ID on stdout.
- help Give a help message and exit, just like "-h".
- literal Take file name arguments literally, do not expand wildcards. This has no effect on Unix where the shell expands wildcards.
- nopugin Skip loading plugins. Implied by -u NONE.
- remote Connect to a Vim server and make it edit the files given in the rest of the arguments. If no server is found a warning is given and the files are edited in the current Vim.
- remote-expr {expr} Connect to a Vim server, evaluate {expr} in it and print the result on stdout.
- remote-send {keys} Connect to a Vim server and send {keys} to it.
- remote-silent As --remote, but without the warning when no server is found.
- remote-wait As --remote, but Vim does not exit until the files have been edited.
- remote-wait-silent As --remote-wait, but without the warning when no server is found.
- serverlist List the names of all Vim servers that can be found.
- servername {name} Use {name} as the server name. Used for the current Vim, unless used with a --remote argument, then it's the name of the server to connect to.

--socketid {id}
 GTK GUI only: Use the GtkPlug mechanism to run gvim in another window.

--startuptime {file}
 During startup write timing messages to the file {fname}.

--version Print version information and exit.

ON-LINE HELP

Type ":help" in **Vim** to get started. Type ":help subject" to get help on a specific subject. For example: ":help ZZ" to get help for the "ZZ" command. Use <Tab> and CTRL-D to complete subjects ("help cmd-line-completion"). Tags are present to jump from one place to another (sort of hypertext links, see ":help"). All documentation files can be viewed in this way, for example ":help syntax.txt".

FILES

/usr/share/vim/vim82/doc/*.txt
 The **Vim** documentation files. Use ":help doc-file-list" to get the complete list.

/usr/share/vim/vim82/doc/tags
 The tags file used for finding information in the documentation files.

/usr/share/vim/vim82/syntax/syntax.vim
 System wide syntax initializations.

/usr/share/vim/vim82/syntax/*.vim
 Syntax files for various languages.

/usr/share/vim/vimrc
 System wide **Vim** initializations.

~/.vimrc Your personal **Vim** initializations.

/usr/share/vim/gvimrc
 System wide gvim initializations.

~/.gvimrc Your personal gvim initializations.

/usr/share/vim/vim82/optwin.vim
 Script used for the ":options" command, a nice way to view and set options.

/usr/share/vim/vim82/menu.vim
 System wide menu initializations for gvim.

/usr/share/vim/vim82/bugreport.vim
 Script to generate a bug report. See ":help bugs".

/usr/share/vim/vim82/filetype.vim
 Script to detect the type of a file by its name. See ":help 'filetype'".

/usr/share/vim/vim82/scripts.vim
 Script to detect the type of a file by its contents. See ":help 'filetype'".

/usr/share/vim/vim82/print/*.ps
 Files used for PostScript printing.

For recent info read the VIM home page:
<URL:<http://www.vim.org/>>

SEE ALSO

[vimtutor\(1\)](#)

AUTHOR

Most of **Vim** was made by Bram Moolenaar, with a lot of help from others. See ":help credits" in **Vim**.
Vim is based on Stevie, worked on by: Tim Thompson, Tony Andrews and G.R. (Fred) Walter. Although hardly any of the original code remains.

BUGS

Probably. See ":help todo" for a list of known problems.

Note that a number of things that may be regarded as bugs by some, are in fact caused by a too-faithful reproduction of Vi's behaviour. And if you think other things are bugs "because Vi does it differently", you should take a closer look at the vi_diff.txt file (or type :help vi_diff.txt when in Vim). Also have a look at the 'compatible' and 'coptions' options.

NAME

vim – Vi IMproved, a programmer's text editor

SYNOPSIS

```
vim [options] [file ..]
vim [options] -
vim [options] -t tag
vim [options] -q [errorfile]

ex
view
gvim gview evim eview
rvim rview rgvim rgview
```

DESCRIPTION

Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences between **Vim** and Vi.

While running **Vim** a lot of help can be obtained from the on-line help system, with the ":help" command. See the ON-LINE HELP section below.

Most often **Vim** is started to edit a single file with the command

```
vim file
```

More generally **Vim** is started with:

```
vim [options] [filelist]
```

If the filelist is missing, the editor will start with an empty buffer. Otherwise exactly one out of the following four may be used to choose one or more files to be edited.

- | | |
|----------------|---|
| file .. | A list of filenames. The first one will be the current file and read into the buffer. The cursor will be positioned on the first line of the buffer. You can get to the other files with the ":next" command. To edit a file that starts with a dash, precede the filelist with "--". |
| - | The file to edit is read from stdin. Commands are read from stderr, which should be a tty. |
| -t {tag} | The file to edit and the initial cursor position depends on a "tag", a sort of goto label. {tag} is looked up in the tags file, the associated file becomes the current file and the associated command is executed. Mostly this is used for C programs, in which case {tag} could be a function name. The effect is that the file containing that function becomes the current file and the cursor is positioned on the start of the function. See ":help tag-commands". |
| -q [errorfile] | Start in quickFix mode. The file [errorfile] is read and the first error is displayed. If [errorfile] is omitted, the filename is obtained from the 'errorfile' option (defaults to "AztecC.Err" for the Amiga, "errors.err" on other systems). Further errors can be jumped to with the ":cn" command. See ":help quickfix". |

Vim behaves differently, depending on the name of the command (the executable may still be the same file).

- | | |
|------|---|
| vim | The "normal" way, everything is default. |
| ex | Start in Ex mode. Go to Normal mode with the ":vi" command. Can also be done with the "-e" argument. |
| view | Start in read-only mode. You will be protected from writing the files. Can also be done with the "-R" argument. |

gvim gview

The GUI version. Starts a new window. Can also be done with the "-g" argument.

evim eview

The GUI version in easy mode. Starts a new window. Can also be done with the "-y" argument.

rvim rview rgvim rgview

Like the above, but with restrictions. It will not be possible to start shell commands, or suspend **Vim**. Can also be done with the "-Z" argument.

OPTIONS

The options may be given in any order, before or after filenames. Options without an argument can be combined after a single dash.

- +[num] For the first file the cursor will be positioned on line "num". If "num" is missing, the cursor will be positioned on the last line.
- +/ {pat} For the first file the cursor will be positioned in the line with the first occurrence of {pat}. See ":help search-pattern" for the available search patterns.
- +{command}
- c {command} {command} will be executed after the first file has been read. {command} is interpreted as an Ex command. If the {command} contains spaces it must be enclosed in double quotes (this depends on the shell that is used). Example: vim "+set si" main.c
Note: You can use up to 10 "+" or "-c" commands.
- S {file} {file} will be sourced after the first file has been read. This is equivalent to -c "source {file} ". {file} cannot start with '-'. If {file} is omitted "Session.vim" is used (only works when -S is the last argument).
- cmd {command} Like using "-c", but the command is executed just before processing any vimrc file. You can use up to 10 of these commands, independently from "-c" commands.
- A If **Vim** has been compiled with ARABIC support for editing right-to-left oriented files and Arabic keyboard mapping, this option starts **Vim** in Arabic mode, i.e. 'arabic' is set. Otherwise an error message is given and **Vim** aborts.
- b Binary mode. A few options will be set that makes it possible to edit a binary or executable file.
- C Compatible. Set the 'compatible' option. This will make **Vim** behave mostly like Vi, even though a .vimrc file exists.
- d Start in diff mode. There should between two to eight file name arguments. **Vim** will open all the files and show differences between them. Works like vimdiff(1).
- d {device} Open {device} for use as a terminal. Only on the Amiga. Example: "-d con:20/30/600/150".
- D Debugging. Go to debugging mode when executing the first command from a script.
- e Start **Vim** in Ex mode, just like the executable was called "ex".
- E Start **Vim** in improved Ex mode, just like the executable was called "exim".
- f Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in. On the Amiga, **Vim** is not restarted to open a new window. This option should be used when **Vim** is executed by a program that will wait for the edit session to finish (e.g. mail). On the Amiga the ":sh" and ":!" commands will not work.
- nofork Foreground. For the GUI version, **Vim** will not fork and detach from the shell it was started in.

- F If **Vim** has been compiled with FKMAP support for editing right-to-left oriented files and Farsi keyboard mapping, this option starts **Vim** in Farsi mode, i.e. 'fkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.
- g If **Vim** has been compiled with GUI support, this option enables the GUI. If no GUI support was compiled in, an error message is given and **Vim** aborts.
- h Give a bit of help about the command line arguments and options. After this **Vim** exits.
- H If **Vim** has been compiled with RIGHTLEFT support for editing right-to-left oriented files and Hebrew keyboard mapping, this option starts **Vim** in Hebrew mode, i.e. 'hkmap' and 'rightleft' are set. Otherwise an error message is given and **Vim** aborts.
- i {viminfo} Specifies the filename to use when reading or writing the viminfo file, instead of the default "~/viminfo". This can also be used to skip the use of the .viminfo file, by giving the name "NONE".
- L Same as -r.
- l Lisp mode. Sets the 'lisp' and 'showmatch' options on.
- m Modifying files is disabled. Resets the 'write' option. You can still modify the buffer, but writing a file is not possible.
- M Modifications not allowed. The 'modifiable' and 'write' options will be unset, so that changes are not allowed and files can not be written. Note that these options can be set to enable making modifications.
- N No-compatible mode. Resets the 'compatible' option. This will make **Vim** behave a bit better, but less Vi compatible, even though a .vimrc file does not exist.
- n No swap file will be used. Recovery after a crash will be impossible. Handy if you want to edit a file on a very slow medium (e.g. floppy). Can also be done with ":set uc=0". Can be undone with ":set uc=200".
- nb Become an editor server for NetBeans. See the docs for details.
- o[N] Open N windows stacked. When N is omitted, open one window for each file.
- O[N] Open N windows side by side. When N is omitted, open one window for each file.
- p[N] Open N tab pages. When N is omitted, open one tab page for each file.
- R Read-only mode. The 'readonly' option will be set. You can still edit the buffer, but will be prevented from accidentally overwriting a file. If you do want to overwrite a file, add an exclamation mark to the Ex command, as in ":w!". The -R option also implies the -n option (see above). The 'readonly' option can be reset with ":set noro". See ":help 'readonly'".
- r List swap files, with information about using them for recovery.
- r {file} Recovery mode. The swap file is used to recover a crashed editing session. The swap file is a file with the same filename as the text file with ".swp" appended. See ":help recovery".
- s Silent mode. Only when started as "Ex" or when the "-e" option was given before the "-s" option.
- s {scriptin} The script file {scriptin} is read. The characters in the file are interpreted as if you had typed them. The same can be done with the command ":source! {scriptin}". If the end of the file is reached before the editor exits, further characters are read from the keyboard.
- T {terminal} Tells **Vim** the name of the terminal you are using. Only required when the automatic way doesn't work. Should be a terminal known to **Vim** (builtin) or defined in the termcap or terminfo file.
- u {vimrc} Use the commands in the file {vimrc} for initializations. All the other initializations are skipped. Use this to edit a special kind of files. It can also be used to skip all initializations

by giving the name "NONE". See ":help initialization" within vim for more details.

- U {gvimrc} Use the commands in the file {gvimrc} for GUI initializations. All the other GUI initializations are skipped. It can also be used to skip all GUI initializations by giving the name "NONE". See ":help gui-init" within vim for more details.
- V[N] Verbose. Give messages about which files are sourced and for reading and writing a viminfo file. The optional number N is the value for 'verbose'. Default is 10.
- v Start **Vim** in Vi mode, just like the executable was called "vi". This only has effect when the executable is called "ex".
- w {scriptout} All the characters that you type are recorded in the file {scriptout}, until you exit **Vim**. This is useful if you want to create a script file to be used with "vim -s" or ":source!". If the {scriptout} file exists, characters are appended.
- W {scriptout} Like -w, but an existing file is overwritten.
- x Use encryption when writing files. Will prompt for a crypt key.
- X Don't connect to the X server. Shortens startup time in a terminal, but the window title and clipboard will not be used.
- y Start **Vim** in easy mode, just like the executable was called "evim" or "eview". Makes **Vim** behave like a click-and-type editor.
- Z Restricted mode. Works like the executable starts with "r".
- Denotes the end of the options. Arguments after this will be handled as a file name. This can be used to edit a filename that starts with a '-'.
- clean Do not use any personal configuration (vimrc, plugins, etc.). Useful to see if a problem reproduces with a clean Vim setup.
- echo-wid GTK GUI only: Echo the Window ID on stdout.
- help Give a help message and exit, just like "-h".
- literal Take file name arguments literally, do not expand wildcards. This has no effect on Unix where the shell expands wildcards.
- nopugin Skip loading plugins. Implied by -u NONE.
- remote Connect to a Vim server and make it edit the files given in the rest of the arguments. If no server is found a warning is given and the files are edited in the current Vim.
- remote-expr {expr} Connect to a Vim server, evaluate {expr} in it and print the result on stdout.
- remote-send {keys} Connect to a Vim server and send {keys} to it.
- remote-silent As --remote, but without the warning when no server is found.
- remote-wait As --remote, but Vim does not exit until the files have been edited.
- remote-wait-silent As --remote-wait, but without the warning when no server is found.
- serverlist List the names of all Vim servers that can be found.
- servername {name} Use {name} as the server name. Used for the current Vim, unless used with a --remote argument, then it's the name of the server to connect to.

--socketid {id}
 GTK GUI only: Use the GtkPlug mechanism to run gvim in another window.
 --startuptime {file}
 During startup write timing messages to the file {fname}.
 --version Print version information and exit.

ON-LINE HELP

Type ":help" in **Vim** to get started. Type ":help subject" to get help on a specific subject. For example: ":help ZZ" to get help for the "ZZ" command. Use <Tab> and CTRL-D to complete subjects ("help cmd-line-completion"). Tags are present to jump from one place to another (sort of hypertext links, see ":help"). All documentation files can be viewed in this way, for example ":help syntax.txt".

FILES

/usr/share/vim/vim82/doc/*.txt
 The **Vim** documentation files. Use ":help doc-file-list" to get the complete list.
 /usr/share/vim/vim82/doc/tags
 The tags file used for finding information in the documentation files.
 /usr/share/vim/vim82/syntax/syntax.vim
 System wide syntax initializations.
 /usr/share/vim/vim82/syntax/*.vim
 Syntax files for various languages.
 /usr/share/vim/vimrc
 System wide **Vim** initializations.
 ~/.vimrc Your personal **Vim** initializations.
 /usr/share/vim/gvimrc
 System wide gvim initializations.
 ~/.gvimrc Your personal gvim initializations.
 /usr/share/vim/vim82/optwin.vim
 Script used for the ":options" command, a nice way to view and set options.
 /usr/share/vim/vim82/menu.vim
 System wide menu initializations for gvim.
 /usr/share/vim/vim82/bugreport.vim
 Script to generate a bug report. See ":help bugs".
 /usr/share/vim/vim82/filetype.vim
 Script to detect the type of a file by its name. See ":help 'filetype'".
 /usr/share/vim/vim82/scripts.vim
 Script to detect the type of a file by its contents. See ":help 'filetype'".
 /usr/share/vim/vim82/print/*.ps
 Files used for PostScript printing.

For recent info read the VIM home page:
<URL:<http://www.vim.org/>>

SEE ALSO

[vimtutor\(1\)](#)

AUTHOR

Most of **Vim** was made by Bram Moolenaar, with a lot of help from others. See ":help credits" in **Vim**.
Vim is based on Stevie, worked on by: Tim Thompson, Tony Andrews and G.R. (Fred) Walter. Although hardly any of the original code remains.

BUGS

Probably. See ":help todo" for a list of known problems.

Note that a number of things that may be regarded as bugs by some, are in fact caused by a too-faithful reproduction of Vi's behaviour. And if you think other things are bugs "because Vi does it differently", you should take a closer look at the vi_diff.txt file (or type :help vi_diff.txt when in Vim). Also have a look at the 'compatible' and 'coptions' options.

NAME

watch – execute a program periodically, showing output fullscreen

SYNOPSIS

watch [*options*] *command*

DESCRIPTION

watch runs *command* repeatedly, displaying its output and errors (the first screenfull). This allows you to watch the program output change over time. By default, *command* is run every 2 seconds and **watch** will run until interrupted.

OPTIONS

-d, --differences[=permanent]

Highlight the differences between successive updates. If the optional *permanent* argument is specified then **watch** will show all changes since the first iteration.

-n, --interval seconds

Specify update interval. The command will not allow quicker than 0.1 second interval, in which the smaller values are converted. Both '.' and ',' work for any locales. The WATCH_INTERVAL environment can be used to persistently set a non-default interval (following the same rules and formatting).

-p, --precise

Make **watch** attempt to run *command* every --interval seconds. Try it with **ntpdate** (if present) and notice how the fractional seconds stays (nearly) the same, as opposed to normal mode where they continuously increase.

-t, --no-title

Turn off the header showing the interval, command, and current time at the top of the display, as well as the following blank line.

-b, --beep

Beep if command has a non-zero exit.

-e, --erexit

Freeze updates on command error, and exit after a key press.

-g, --chgexit

Exit when the output of *command* changes.

-c, --color

Interpret ANSI color and style sequences.

-x, --exec

Pass *command* to **exec(2)** instead of **sh -c** which reduces the need to use extra quoting to get the desired effect.

-w, --no-linewrap

Turn off line wrapping. Long lines will be truncated instead of wrapped to the next line.

-h, --help

Display help text and exit.

-v, --version

Display version information and exit.

EXIT STATUS

0 Success.

1 Various failures.

2 Forking the process to watch failed.

3 Replacing child process stdout with write side pipe failed.

4 Command execution failed.

- 5 Closing child process write pipe failed.
- 7 IPC pipe creation failed.
- 8 Getting child process return value with `waitpid(2)` failed, or command exited up on error.
- other The watch will propagate command exit status as child exit status.

ENVIRONMENT

The behaviour of **watch** is affected by the following environment variables.

WATCH_INTERVAL

Update interval, follows the same rules as the `--interval` command line option.

NOTES

POSIX option processing is used (i.e., option processing stops at the first non-option argument). This means that flags after *command* don't get interpreted by **watch** itself.

BUGS

Upon terminal resize, the screen will not be correctly repainted until the next scheduled update. All `--differences` highlighting is lost on that update as well.

Non-printing characters are stripped from program output. Use `cat -v` as part of the command pipeline if you want to see them.

Combining Characters that are supposed to display on the character at the last column on the screen may display one column early, or they may not display at all.

Combining Characters never count as different in `--differences` mode. Only the base character counts.

Blank lines directly after a line which ends in the last column do not display.

`--precise` mode doesn't yet have advanced temporal distortion technology to compensate for a *command* that takes more than `--interval` seconds to execute. **watch** also can get into a state where it rapid-fires as many executions of *command* as it can to catch up from a previous executions running longer than `--interval` (for example, `netstat` taking ages on a DNS lookup).

EXAMPLES

To watch for mail, you might do

```
watch -n 60 from
```

To watch the contents of a directory change, you could use

```
watch -d ls -l
```

If you're only interested in files owned by user joe, you might use

```
watch -d 'ls -l | fgrep joe'
```

To see the effects of quoting, try these out

```
watch echo $$  
watch echo '$$'  
watch echo "$$"$$"
```

To see the effect of precision time keeping, try adding `-p` to

```
watch -n 10 sleep 1
```

You can watch for your administrator to install the latest kernel with

```
watch uname -r
```

(Note that `-p` isn't guaranteed to work across reboots, especially in the face of `ntpdate` (if present) or other bootup time-changing mechanisms)

NAME

wc – print newline, word, and byte counts for each file

SYNOPSIS

wc [*OPTION*]... [*FILE*]...
wc [*OPTION*]... --files0-from=*F*

DESCRIPTION

Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. A word is a non-zero-length sequence of characters delimited by white space.

With no FILE, or when FILE is **-**, read standard input.

The options below may be used to select which counts are printed, always in the following order: newline, word, character, byte, maximum line length.

-c, --bytes

print the byte counts

-m, --chars

print the character counts

-l, --lines

print the newline counts

--files0-from=*F*

read input from the files specified by NUL-terminated names in file *F*; If *F* is **-** then read names from standard input

-L, --max-line-length

print the maximum display width

-w, --words

print the word counts

--help display this help and exit**--version**

output version information and exit

AUTHOR

Written by Paul Rubin and David MacKenzie.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/wc>>
or available locally via: info '(coreutils) wc invocation'

NAME

whatis – display one-line manual page descriptions

SYNOPSIS

```
whatis [ -dlv?V ] [ -r | -w ] [ -s list ] [ -m system [ ,... ] ] [ -M path ] [ -L locale ] [ -C file ] name ...
```

DESCRIPTION

Each manual page has a short description available within it. **whatis** searches the manual page names and displays the manual page descriptions of any *name* matched.

name may contain wildcards (**-w**) or be a regular expression (**-r**). Using these options, it may be necessary to quote the *name* or escape (\) the special characters to stop the shell from interpreting them.

index databases are used during the search, and are updated by the **mandb** program. Depending on your installation, this may be run by a periodic cron job, or may need to be run manually after new manual pages have been installed. To produce an old style text **whatis** database from the relative **index** database, issue the command:

```
whatis -M manpath -w '*' | sort > manpath/whatis
```

where *manpath* is a manual page hierarchy such as */usr/man*.

OPTIONS**-d, --debug**

Print debugging information.

-v, --verbose

Print verbose warning messages.

-r, --regex

Interpret each *name* as a regular expression. If *aname* matches any part of a page name, a match will be made. This option causes **whatis** to be somewhat slower due to the nature of database searches.

-w, --wildcard

Interpret each *name* as a pattern containing shell style wildcards. For a match to be made, an expanded *name* must match the entire page name. This option causes **whatis** to be somewhat slower due to the nature of database searches.

-l, --long

Do not trim output to the terminal width. Normally, output will be truncated to the terminal width to avoid ugly results from poorly-written **NAME** sections.

-s *list*, --sections=*list*, --section=*list*

Search only the given manual sections. *list* is a colon- or comma-separated list of sections. If an entry in *list* is a simple section, for example "3", then the displayed list of descriptions will include pages in sections "3", "3perl", "3x", and so on; while if an entry in *list* has an extension, for example "3perl", then the list will only include pages in that exact part of the manual section.

-m *system* [,...], --systems=*system* [,...]

If this system has access to other operating systems' manual page names, they can be accessed using this option. To search NewOS's manual page names, use the option **-m NewOS**.

The *system* specified can be a combination of comma delimited operating system names. To include a search of the native operating system's manual page names, include the system name **man** in the argument string. This option will override the **\$SYSTEM** environment variable.

-M *path*, --manpath=*path*

Specify an alternate set of colon-delimited manual page hierarchies to search. By default, **whatis** uses the **\$MANPATH** environment variable, unless it is empty or unset, in which case it will

determine an appropriate manpath based on your **\$PATH** environment variable. This option overrides the contents of **\$MANPATH**.

-L *locale*, **--locale**=*locale*

whatis will normally determine your current locale by a call to the C function **setlocale(3)** which interrogates various environment variables, possibly including **\$LC_MESSAGES** and **\$LANG**. To temporarily override the determined value, use this option to supply a *locale* string directly to **whatis**. Note that it will not take effect until the search for pages actually begins. Output such as the help message will always be displayed in the initially determined locale.

-C *file*, **--config-file**=*file*

Use this user configuration file rather than the default of **~/.manpath**.

-?, --help

Print a help message and exit.

--usage

Print a short usage message and exit.

-V, --version

Display version information.

EXIT STATUS

- 0** Successful program execution.
- 1** Usage, syntax or configuration file error.
- 2** Operational error.
- 16** Nothing was found that matched the criteria specified.

ENVIRONMENT

SYSTEM

If **\$SYSTEM** is set, it will have the same effect as if it had been specified as the argument to the **-m** option.

MANPATH

If **\$MANPATH** is set, its value is interpreted as the colon-delimited manual page hierarchy search path to use.

See the **SEARCH PATH** section of **manpath(5)** for the default behaviour and details of how this environment variable is handled.

MANWIDTH

If **\$MANWIDTH** is set, its value is used as the terminal width (see the **--long** option). If it is not set, the terminal width will be calculated using the value of **\$COLUMNS**, and **ioctl(2)** if available, or falling back to 80 characters if all else fails.

FILES

/usr/share/man/index.(bt/db/dir/pag)

A traditional global *index* database cache.

/var/cache/man/index.(bt/db/dir/pag)

An FHS compliant global *index* database cache.

/usr/share/man/.../whatis

A traditional **whatis** text database.

SEE ALSO

apropos(1), man(1), mandb(8)

AUTHOR

Wilf. (G.Wilford@ee.surrey.ac.uk).

Fabrizio Polacco (fpolacco@debian.org).

Colin Watson (cjwatson@debian.org).

BUGS

<https://gitlab.com/cjwatson/man-db/-/issues>

<https://savannah.nongnu.org/bugs/?group=man-db>

NAME

whereis – locate the binary, source, and manual page files for a command

SYNOPSIS

whereis [options] [**-BMS** *directory...* **-f**] *name...*

DESCRIPTION

whereis locates the binary, source and manual files for the specified command names. The supplied names are first stripped of leading pathname components. Prefixes of **s**, resulting from use of source code control are also dealt with. **whereis** then attempts to locate the desired program in the standard Linux places, and in the places specified by **\$PATH** and **\$MANPATH**.

The search restrictions (options **-b**, **-m** and **-s**) are cumulative and apply to the subsequent *name* patterns on the command line. Any new search restriction resets the search mask. For example,

whereis -bm ls tr -m gcc

searches for "ls" and "tr" binaries and man pages, and for "gcc" man pages only.

The options **-B**, **-M** and **-S** reset search paths for the subsequent *name* patterns. For example,

whereis -m ls -M /usr/share/man/man1 -f cal

searches for "ls" man pages in all default paths, but for "cal" in the */usr/share/man/man1* directory only.

OPTIONS**-b**

Search for binaries.

-m

Search for manuals.

-s

Search for sources.

-u

Only show the command names that have unusual entries. A command is said to be unusual if it does not have just one entry of each explicitly requested type. Thus '**whereis -m -u ***' asks for those files in the current directory which have no documentation file, or more than one.

-B *list*

Limit the places where **whereis** searches for binaries, by a whitespace-separated list of directories.

-M *list*

Limit the places where **whereis** searches for manuals and documentation in Info format, by a whitespace-separated list of directories.

-S *list*

Limit the places where **whereis** searches for sources, by a whitespace-separated list of directories.

-f

Terminates the directory list and signals the start of filenames. It *must* be used when any of the **-B**, **-M**, or **-S** options is used.

-l

Output the list of effective lookup paths that **whereis** is using. When none of **-B**, **-M**, or **-S** is

specified, the option will output the hard-coded paths that the command was able to find on the system.

-h, --help

Display help text and exit.

-V, --version

Print version and exit.

FILE SEARCH PATHS

By default **whereis** tries to find files from hard-coded paths, which are defined with glob patterns. The command attempts to use the contents of **\$PATH** and **\$MANPATH** environment variables as default search path. The easiest way to know what paths are in use is to add the **-l** listing option. Effects of the **-B**, **-M**, and **-S** are displayed with **-l**.

ENVIRONMENT**WHEREIS_DEBUG=all**

enables debug output.

EXAMPLES

To find all files in */usr/bin* which are not documented in */usr/man/man1* or have no source in */usr/src*:

```
cd /usr/bin whereis -u -ms -M /usr/man/man1 -S /usr/src -f *
```

REPORTING BUGS

For bug reports, use the issue tracker at <https://github.com/util-linux/util-linux/issues>.

AVAILABILITY

The **whereis** command is part of the util-linux package which can be downloaded from [Linux Kernel Archive](https://www.kernel.org/pub/linux/utils/util-linux/) <<https://www.kernel.org/pub/linux/utils/util-linux/>>.

NAME

who – show who is logged on

SYNOPSIS

who [*OPTION*]... [*FILE* / *ARG1 ARG2*]

DESCRIPTION

Print information about users who are currently logged in.

-a, --all

same as **-b -d --login -p -r -t -T -u**

-b, --boot

time of last system boot

-d, --dead

print dead processes

-H, --heading

print line of column headings

--ips print ips instead of hostnames. with **--lookup**, canonicalizes based on stored IP, if available, rather than stored hostname

-l, --login

print system login processes

--lookup

attempt to canonicalize hostnames via DNS

-m only hostname and user associated with stdin

-p, --process

print active processes spawned by init

-q, --count

all login names and number of users logged on

-r, --runlevel

print current runlevel

-s, --short

print only name, line, and time (default)

-t, --time

print last system clock change

-T, -w, --mesg

add user's message status as +, - or ?

-u, --users

list users logged in

--message

same as **-T**

--writable

same as **-T**

--help display this help and exit

--version

output version information and exit

If FILE is not specified, use */var/run/utmp*. */var/log/wtmp* as FILE is common. If ARG1 ARG2 given, **-m** presumed: 'am i' or 'mom likes' are usual.

AUTHOR

Written by Joseph Arceneaux, David MacKenzie, and Michael Stone.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>
Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/who>>
or available locally via: info '(coreutils) who invocation'

NAME

whoami – print effective userid

SYNOPSIS

whoami [*OPTION*]...

DESCRIPTION

Print the user name associated with the current effective user ID. Same as id **-un**.

--help display this help and exit

--version

output version information and exit

AUTHOR

Written by Richard Mlynarik.

REPORTING BUGS

GNU coreutils online help: <<https://www.gnu.org/software/coreutils/>>

Report any translation bugs to <<https://translationproject.org/team/>>

COPYRIGHT

Copyright © 2020 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later
<<https://gnu.org/licenses/gpl.html>>.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO

Full documentation <<https://www.gnu.org/software/coreutils/whoami>>
or available locally via: info '(coreutils) whoami invocation'

NAME

xattr – Extended attributes

DESCRIPTION

Extended attributes are name:value pairs associated permanently with files and directories, similar to the environment strings associated with a process. An attribute may be defined or undefined. If it is defined, its value may be empty or non-empty.

Extended attributes are extensions to the normal attributes which are associated with all inodes in the system (i.e., the `stat(2)` data). They are often used to provide additional functionality to a filesystem—for example, additional security features such as Access Control Lists (ACLs) may be implemented using extended attributes.

Users with search access to a file or directory may use `listxattr(2)` to retrieve a list of attribute names defined for that file or directory.

Extended attributes are accessed as atomic objects. Reading (`getxattr(2)`) retrieves the whole value of an attribute and stores it in a buffer. Writing (`setxattr(2)`) replaces any previous value with the new value.

Space consumed for extended attributes may be counted towards the disk quotas of the file owner and file group.

Extended attribute namespaces

Attribute names are null-terminated strings. The attribute name is always specified in the fully qualified *namespace.attribute* form, for example, `user.mime_type`, `trusted.md5sum`, `system.posix_acl_access`, or `security.selinux`.

The namespace mechanism is used to define different classes of extended attributes. These different classes exist for several reasons; for example, the permissions and capabilities required for manipulating extended attributes of one namespace may differ to another.

Currently, the `security`, `system`, `trusted`, and `user` extended attribute classes are defined as described below. Additional classes may be added in the future.

Extended security attributes

The security attribute namespace is used by kernel security modules, such as Security Enhanced Linux, and also to implement file capabilities (see `capabilities(7)`). Read and write access permissions to security attributes depend on the policy implemented for each security attribute by the security module. When no security module is loaded, all processes have read access to extended security attributes, and write access is limited to processes that have the `CAP_SYS_ADMIN` capability.

System extended attributes

System extended attributes are used by the kernel to store system objects such as Access Control Lists. Read and write access permissions to system attributes depend on the policy implemented for each system attribute implemented by filesystems in the kernel.

Trusted extended attributes

Trusted extended attributes are visible and accessible only to processes that have the `CAP_SYS_ADMIN` capability. Attributes in this class are used to implement mechanisms in user space (i.e., outside the kernel) which keep information in extended attributes to which ordinary processes should not have access.

User extended attributes

User extended attributes may be assigned to files and directories for storing arbitrary additional information such as the mime type, character set or encoding of a file. The access permissions for user attributes are defined by the file permission bits: read permission is required to retrieve the attribute value, and writer permission is required to change it.

The file permission bits of regular files and directories are interpreted differently from the file permission bits of special files and symbolic links. For regular files and directories the file permission bits define access to the file's contents, while for device special files they define access to the device described by the special file. The file permissions of symbolic links are not used in access checks. These differences would allow users to consume filesystem resources in a way not controllable by disk quotas for group or world

writable special files and directories.

For this reason, user extended attributes are allowed only for regular files and directories, and access to user extended attributes is restricted to the owner and to users with appropriate capabilities for directories with the sticky bit set (see the **chmod(1)** manual page for an explanation of the sticky bit).

Filesystem differences

The kernel and the filesystem may place limits on the maximum number and size of extended attributes that can be associated with a file. The VFS-imposed limits on attribute names and values are 255 bytes and 64 kB, respectively. The list of attribute names that can be returned is also limited to 64 kB (see BUGS in **listxattr(2)**).

Some filesystems, such as Reiserfs (and, historically, ext2 and ext3), require the filesystem to be mounted with the **user_xattr** mount option in order for user extended attributes to be used.

In the current ext2, ext3, and ext4 filesystem implementations, the total bytes used by the names and values of all of a file's extended attributes must fit in a single filesystem block (1024, 2048 or 4096 bytes, depending on the block size specified when the filesystem was created).

In the Btrfs, XFS, and Reiserfs filesystem implementations, there is no practical limit on the number of extended attributes associated with a file, and the algorithms used to store extended attribute information on disk are scalable.

In the JFS, XFS, and Reiserfs filesystem implementations, the limit on bytes used in an EA value is the ceiling imposed by the VFS.

In the Btrfs filesystem implementation, the total bytes used for the name, value, and implementation overhead bytes is limited to the filesystem *nodesize* value (16 kB by default).

STANDARDS

Extended attributes are not specified in POSIX.1, but some other systems (e.g., the BSDs and Solaris) provide a similar feature.

NOTES

Since the filesystems on which extended attributes are stored might also be used on architectures with a different byte order and machine word size, care should be taken to store attribute values in an architecture-independent format.

This page was formerly named **attr(5)**.

SEE ALSO

attr(1), **getfattr(1)**, **setfattr(1)**, **getxattr(2)**, **ioctl_iflags(2)**, **listxattr(2)**, **removexattr(2)**, **setxattr(2)**, **acl(5)**, **capabilities(7)**, **selinux(8)**

NAME

xfs – layout, mount options, and supported file attributes for the XFS filesystem

DESCRIPTION

An XFS filesystem can reside on a regular disk partition or on a logical volume. An XFS filesystem has up to three parts: a data section, a log section, and a realtime section. Using the default **mkfs.xfs(8)** options, the realtime section is absent, and the log area is contained within the data section. The log section can be either separate from the data section or contained within it. The filesystem sections are divided into a certain number of *blocks*, whose size is specified at **mkfs.xfs(8)** time with the **-b** option.

The data section contains all the filesystem metadata (inodes, directories, indirect blocks) as well as the user file data for ordinary (non-realtime) files and the log area if the log is *internal* to the data section. The data section is divided into a number of *allocation groups*. The number and size of the allocation groups are chosen by **mkfs.xfs(8)** so that there is normally a small number of equal-sized groups. The number of allocation groups controls the amount of parallelism available in file and block allocation. It should be increased from the default if there is sufficient memory and a lot of allocation activity. The number of allocation groups should not be set very high, since this can cause large amounts of CPU time to be used by the filesystem, especially when the filesystem is nearly full. More allocation groups are added (of the original size) when **xfs_growfs(8)** is run.

The log section (or area, if it is internal to the data section) is used to store changes to filesystem metadata while the filesystem is running until those changes are made to the data section. It is written sequentially during normal operation and read only during mount. When mounting a filesystem after a crash, the log is read to complete operations that were in progress at the time of the crash.

The realtime section is used to store the data of realtime files. These files had an attribute bit set through **xfctl(3)** after file creation, before any data was written to the file. The realtime section is divided into a number of *extents* of fixed size (specified at **mkfs.xfs(8)** time). Each file in the realtime section has an extent size that is a multiple of the realtime section extent size.

Each allocation group contains several data structures. The first sector contains the superblock. For allocation groups after the first, the superblock is just a copy and is not updated after **mkfs.xfs(8)**. The next three sectors contain information for block and inode allocation within the allocation group. Also contained within each allocation group are data structures to locate free blocks and inodes; these are located through the header structures.

Each XFS filesystem is labeled with a Universal Unique Identifier (UUID). The UUID is stored in every allocation group header and is used to help distinguish one XFS filesystem from another, therefore you should avoid using **dd(1)** or other block-by-block copying programs to copy XFS filesystems. If two XFS filesystems on the same machine have the same UUID, **xfsdump(8)** may become confused when doing incremental and resumed dumps. **xfsdump(8)** and **xfs_r estore(8)** are recommended for making copies of XFS filesystems.

OPERATIONS

Some functionality specific to the XFS filesystem is accessible to applications through the **xfctl(3)** and by-handle (see **open_by_handle(3)**) interfaces.

MOUNT OPTIONS

The following XFS-specific mount options may be used when mounting an XFS filesystem. Other generic options may be used as well; refer to the **mount(8)** manual page for more details.

allocsize=size

Sets the buffered I/O end-of-file preallocation size when doing delayed allocation writeout. Valid values for this option are page size (typically 4KiB) through to 1GiB, inclusive, in power-of-2 increments.

The default behavior is for dynamic end-of-file preallocation size, which uses a set of heuristics to optimise the preallocation size based on the current allocation patterns within the file and the access patterns to the file. Specifying a fixed allocsize value turns off the dynamic behavior.

attr2|noattr2

Note: These options have been **deprecated** as of kernel v5.10; The noattr2 option will be removed no earlier than in September 2025 and attr2 option will be immutable default.

The options enable/disable an "opportunistic" improvement to be made in the way inline extended attributes are stored on-disk. When the new form is used for the first time when attr2 is selected (either when setting or removing extended attributes) the on-disk superblock feature bit field will be updated to reflect this format being in use.

The default behavior is determined by the on-disk feature bit indicating that attr2 behavior is active. If either mount option is set, then that becomes the new default used by the filesystem.

CRC enabled filesystems always use the attr2 format, and so will reject the noattr2 mount option if it is set.

dax=value

Set CPU direct access (DAX) behavior for the current filesystem. This mount option accepts the following values:

"dax=inode" DAX will be enabled only on regular files with FS_XFLAG_DAX applied.

"dax=never" DAX will not be enabled for any files. FS_XFLAG_DAX will be ignored.

"dax=always" DAX will be enabled for all regular files, regardless of the FS_XFLAG_DAX state.

If no option is used when mounting a filesystem stored on a DAX capable device, dax=inode will be used as default.

For details regarding DAX behavior in kernel, please refer to kernel's documentation at filesystems/dax.txt

discard|nodiscard

Enable/disable the issuing of commands to let the block device reclaim space freed by the filesystem. This is useful for SSD devices, thinly provisioned LUNs and virtual machine images, but may have a performance impact.

Note: It is currently recommended that you use the fstrim application to discard unused blocks rather than the discard mount option because the performance impact of this option is quite severe. For this reason, nodiscard is the default.

grpid|bsdgroups|nogrp|sysvgroups

These options define what group ID a newly created file gets. When grpid is set, it takes the group ID of the directory in which it is created; otherwise it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

filestreams

Make the data allocator use the filestreams allocation mode across the entire filesystem rather than just on directories configured to use it.

ikeep|noikeep

Note: These options have been **deprecated** as of kernel v5.10; The noikeep option will be removed no earlier than in September 2025 and ikeep option will be immutable default.

When ikeep is specified, XFS does not delete empty inode clusters and keeps them around on disk. When noikeep is specified, empty inode clusters are returned to the free space pool. noikeep is the

default.

inode32|inode64

When inode32 is specified, it indicates that XFS limits inode creation to locations which will not result in inode numbers with more than 32 bits of significance.

When inode64 is specified, it indicates that XFS is allowed to create inodes at any location in the filesystem, including those which will result in inode numbers occupying more than 32 bits of significance.

inode32 is provided for backwards compatibility with older systems and applications, since 64 bits inode numbers might cause problems for some applications that cannot handle large inode numbers. If applications are in use which do not handle inode numbers bigger than 32 bits, the inode32 option should be specified.

For kernel v3.7 and later, inode64 is the default.

largeio|nolargeio

If "nolargeio" is specified, the optimal I/O reported in st_blksize by stat(2) will be as small as possible to allow user applications to avoid inefficient read/modify/write I/O. This is typically the page size of the machine, as this is the granularity of the page cache.

If "largeio" specified, a filesystem that was created with a "swidth" specified will return the "swidth" value (in bytes) in st_blksize. If the filesystem does not have a "swidth" specified but does specify an "allocsize" then "allocsize" (in bytes) will be returned instead. Otherwise the behavior is the same as if "nolargeio" was specified. nolargeio is the default.

logbufs=value

Set the number of in-memory log buffers. Valid numbers range from 2–8 inclusive.

The default value is 8 buffers.

If the memory cost of 8 log buffers is too high on small systems, then it may be reduced at some cost to performance on metadata intensive workloads. The logsize option below controls the size of each buffer and so is also relevant to this case.

logsize=value

Set the size of each in-memory log buffer. The size may be specified in bytes, or in kibibytes (KiB) with a "k" suffix. Valid sizes for version 1 and version 2 logs are 16384 (value=16k) and 32768 (value=32k). Valid sizes for version 2 logs also include 65536 (value=64k), 131072 (value=128k) and 262144 (value=256k). The logsize must be an integer multiple of the log stripe unit configured at mkfs time.

The default value for version 1 logs is 32768, while the default value for version 2 logs is max(32768, log_sunit).

logdev=device and rtdev=device

Use an external log (metadata journal) and/or real-time device. An XFS filesystem has up to three parts: a data section, a log section, and a real-time section. The real-time section is optional, and the log section can be separate from the data section or contained within it.

noalign

Data allocations will not be aligned at stripe unit boundaries. This is only relevant to filesystems created with non-zero data alignment parameters (sunit, swidth) by mkfs.

norecovery

The filesystem will be mounted without running log recovery. If the filesystem was not cleanly unmounted, it is likely to be inconsistent when mounted in "norecovery" mode. Some files or directories may not be accessible because of this. Filesystems mounted "norecovery" must be

mounted read-only or the mount will fail.

nouuid Don't check for double mounted file systems using the file system uuid. This is useful to mount LVM snapshot volumes, and often used in combination with "norecovery" for mounting read-only snapshots.

noquota

Forcibly turns off all quota accounting and enforcement within the filesystem.

uquota/usrquota/quota/uqnoenforce/qnoenforce

User disk quota accounting enabled, and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

gquota/grpquota/gqnoenforce

Group disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

pquota/prjquota/pqnoenforce

Project disk quota accounting enabled and limits (optionally) enforced. Refer to xfs_quota(8) for further details.

sunit=value and swidth=value

Used to specify the stripe unit and width for a RAID device or a stripe volume. "value" must be specified in 512-byte block units. These options are only relevant to filesystems that were created with non-zero data alignment parameters.

The sunit and swidth parameters specified must be compatible with the existing filesystem alignment characteristics. In general, that means the only valid changes to sunit are increasing it by a power-of-2 multiple. Valid swidth values are any integer multiple of a valid sunit value.

Typically the only time these mount options are necessary if after an underlying RAID device has had its geometry modified, such as adding a new disk to a RAID5 lun and reshaping it.

swalloc

Data allocations will be rounded up to stripe width boundaries when the current end of file is being extended and the file size is larger than the stripe width size.

wsync When specified, all filesystem namespace operations are executed synchronously. This ensures that when the namespace operation (create, unlink, etc) completes, the change to the namespace is on stable storage. This is useful in HA setups where failover must not result in clients seeing inconsistent namespace presentation during or after a failover event.

REMOVED MOUNT OPTIONS

The following mount options have been removed from the kernel, and will yield mount failures if specified. Mount options are deprecated for a significant period time prior to removal.

Name	Removed
---	-----
delaylog/nodelaylog	v4.0
ihashsize	v4.0
irixsgid	v4.0
osyncisdsync/osyncisosync	v4.0
barrier/nobarrier	v4.19

FILE ATTRIBUTES

The XFS filesystem supports setting the following file attributes on Linux systems using the chattr(1) utility:

a - append only

A - no atime updates

d - no dump

i - immutable

S - synchronous updates

For descriptions of these attribute flags, please refer to the **chattr(1)** man page.

SEE ALSO

chattr(1), **xfsctl(3)**, **mount(8)**, **mkfs.xfs(8)**, **xfs_info(8)**, **xfs_admin(8)**, **xfsdump(8)**, **xfsrestore(8)**.

NAME

xkill - kill a client by its X resource

SYNOPSIS

xkill [**-display** *displayname*] [**-id** *resource*] [**-button** *number*] [**-frame**] [**-all**] [**-version**]

DESCRIPTION

Xkill is a utility for forcing the X server to close connections to clients. This program is very dangerous, but is useful for aborting programs that have displayed undesired windows on a user's screen. If no resource identifier is given with **-id**, *xkill* will display a special cursor as a prompt for the user to select a window to be killed. If a pointer button is pressed over a non-root window, the server will close its connection to the client that created the window.

OPTIONS

-display *displayname*

This option specifies the name of the X server to contact.

-id *resource*

This option specifies the X identifier for the resource whose creator is to be aborted. If no resource is specified, *xkill* will display a special cursor with which you should select a window to be kill.

-button *number*

This option specifies the number of pointer button that should be used in selecting a window to kill. If the word "any" is specified, any button on the pointer may be used. By default, the first button in the pointer map (which is usually the leftmost button) is used.

-all This option indicates that all clients with top-level windows on the screen should be killed. *Xkill* will ask you to select the root window with each of the currently defined buttons to give you several chances to abort. Use of this option is highly discouraged.

-frame This option indicates that *xkill* should ignore the standard conventions for finding top-level client windows (which are typically nested inside a window manager window), and simply believe that you want to kill direct children of the root.

-version

This option makes *xkill* print its version and exit without killing anything.

CAVEATS

This command does not provide any warranty that the application whose connection to the X server is closed will abort nicely, or even abort at all. All this command does is to close the connection to the X server. Many existing applications do indeed abort when their connection to the X server is closed, but some can choose to continue.

XDEFUALTS

Button Specifies a specific pointer button number or the word "any" to use when selecting windows.

SEE ALSO

X(7), xwininfo(1), XKillClient(3), XGetPointerMapping(3), KillClient in the X Protocol Specification

AUTHOR

Jim Fulton, MIT X Consortium

Dana Chee, Bellcore

yum(8) - Linux man page

Name

yum - Yellowdog Updater Modified

Synopsis

yum [options] [command] [package ...]

Description

yum is an interactive, rpm based, package manager. It can automatically perform system updates, including dependency analysis and obsolete processing based on "repository" metadata. It can also perform installation of new packages, removal of old packages and perform queries on the installed and/or available packages among many other commands/services (see below). **yum** is similar to other high level package managers like apt-get and smart.

While there are some graphical interfaces directly to the **yum** code, more recent graphical interface development is happening with PackageKit and the gnome-packagekit application.

command is one of:

- * install package1 [package2] [...]
- * update [package1] [package2] [...]
- * update-to [package1] [package2] [...]
- * check-update
- * upgrade [package1] [package2] [...]
- * upgrade-to [package1] [package2] [...]
- * distribution-synchronization [package1] [package2] [...]
- * remove | erase package1 [package2] [...]
- * list [...]
- * info [...]
- * provides | whatprovides feature1 [feature2] [...]
- * clean [packages | metadata | expire-cache | rpmdb | plugins | all]
- * makecache
- * groupinstall group1 [group2] [...]
- * groupupdate group1 [group2] [...]
- * grouplist [hidden] [groupwildcard] [...]
- * groupremove group1 [group2] [...]
- * groupinfo group1 [...]
- * search string1 [string2] [...]
- * shell [filename]
- * resolvedep dep1 [dep2] [...]
- * localinstall rpmfile1 [rpmfile2] [...] (maintained for legacy reasons only - use install)
- * localupdate rpmfile1 [rpmfile2] [...] (maintained for legacy reasons only - use update)
- * reinstall package1 [package2] [...]
- * downgrade package1 [package2] [...]
- * deplist package1 [package2] [...]

```
* repolist [all|enabled|disabled]
* version [ all | installed | available | group-* | nogroups* | grouplist | groupinfo ]
* history [info|list|packages-list|packages-info|summary|addon-
info|redo|undo|rollback|new|sync|stats]
* load-transaction [txfile]
* check
* help [command]
```

Unless the --help or -h option is given, one of the above commands must be present.

Repository configuration is honored in all operations.

install

Is used to install the latest version of a package or group of packages while ensuring that all dependencies are satisfied. (See **Specifying package names** for more information) If no package matches the given package **name(s)**, they are assumed to be a shell glob and any matches are then installed. If the name starts with an @ character the rest of the name is used as though passed to the groupinstall command. If the name starts with a - character, then a search is done within the transaction and any matches are removed. If the name is a file, then install works like localinstall. If the name doesn't match a package, then package "provides" are searched (e.g. "_sqlitecache.so()(64bit)") as are filelists (Eg. "/usr/bin/yum"). Also note that for filelists, wildcards will match multiple packages.

update

If run without any packages, update will update every currently installed package. If one or more packages or package globs are specified, Yum will only update the listed packages. While updating packages, **yum** will ensure that all dependencies are satisfied. (See **Specifying package names** for more information) If the packages or globs specified match to packages which are not currently installed then update will not install them. update operates on groups, files, provides and filelists just like the "install" command.

If the main obsoletes configure option is true (default) or the --obsoletes flag is present **yum** will include package obsoletes in its calculations - this makes it better for distro-version changes, for example: upgrading from somelinux 8.0 to somelinux 9.

Note that "**update**" works on installed packages first, and only if there are no matches does it look for available packages. The difference is most noticeable when you do "**update** foo-1-2" which will act exactly as "**update** foo" if foo-1-2 is installed. You can use the "**update-to**" if you'd prefer that nothing happen in the above case.

update-to

This command works like "**update**" but always specifies the version of the package we want to update to.

check-update

Implemented so you could know if your machine had any updates that needed to be applied without running it interactively. Returns exit value of 100 if there are packages available for an update. Also returns a list of the packages to be updated in list format. Returns 0 if no packages are available for update. Returns 1 if an error occurred. Running in verbose mode also shows obsoletes.

upgrade

Is the same as the update command with the --obsoletes flag set. See update for more details.

upgrade-to

This command works like "**upgrade**" but always specifies the version of the package we want to update to.

distribution-synchronization or distro-sync

Synchronizes the installed package set with the latest packages available, this is done by either obsoleting, upgrading or downgrading as appropriate. This will "normally" do the same thing as the upgrade command however if you have the package FOO installed at version 4, and the latest available is only version 3, then this command will **downgrade** FOO to version 3.

This command does not perform operations on groups, local packages or negative selections.

remove or erase

Are used to remove the specified packages from the system as well as removing any packages which depend on the package being removed. remove operates on groups, files, provides and filelists just like the "install" command.(See **Specifying package names** for more information)

Note that "yum" is included in the protected_packages configuration, by default. So you can't accidentally remove yum itself.

list

Is used to list various information about available packages; more complete details are available in the *List Options* section below.

provides or whatprovides

Is used to find out which package provides some feature or file. Just use a specific name or a file-glob-syntax wildcards to list the packages available or installed that provide that feature or file.

search

This is used to find packages when you know something about the package but aren't sure of it's name. By default search will try searching just package names and summaries, but if that "fails" it will then try descriptions and url.

Yum search orders the results so that those packages matching more terms will appear first.

You can force searching everything by specifying "all" as the first argument.

info

Is used to list a description and summary information about available packages; takes the same arguments as in the *List Options* section below.

clean

Is used to clean up various things which accumulate in the **yum** cache directory over time. More complete details can be found in the *Clean Options* section below.

makecache

Is used to download and make usable all the metadata for the currently enabled **yum** repos.

groupinstall

Is used to install all of the individual packages in a group, of the specified types (this works as if you'd taken each of those package names and put them on the command line for a "yum install" command). The group_package_types configuration option specifies which types will be installed.

groupupdate

Is just an alias for groupinstall, which will do the right thing because "yum install X" and "yum update X" do the same thing, when X is already installed.

grouplist

Is used to list the available groups from all **yum** repos. Groups are marked as "installed" if all mandatory packages are installed, or if a group doesn't have any mandatory packages then it is installed if any of the optional or default package are installed. The optional "hidden" argument will also list groups marked as not being "user visible". If you pass the -v option, to enable verbose mode, then the groupids are displayed.

groupremove

Is used to remove all of the packages in a group, unlike "groupinstall" this will remove everything regardless of group_package_types. It is worth pointing out that packages can be in more than one group, so "groupinstall X Y" followed by "groupremove Y" does not do give you the same result as "groupinstall X".

The groupremove_leaf_only configuration changes the behaviour of this command to only remove packages which aren't required by something else.

groupinfo

Is used to give the description and package list of a group (and which type those packages are marked as). Note that you can use the yum-filter-data and yum-list-data plugins to get/use the data the other way around (Ie. what groups own packages need updating). If you pass the -v option, to enable verbose mode, then the package names are matched against installed/available packages similar to the list command.

shell

Is used to enter the 'yum shell', when a filename is specified the contents of that file is executed in yum shell mode. See **yum-shell(8)** for more info

resolvedep

Is used to list packages providing the specified dependencies, at most one package is listed per dependency.

localinstall

Is used to install a set of local rpm files. If required the enabled repositories will be used to resolve dependencies. Note that the install command will do a local install, if given a filename. This option is maintained for legacy reasons only.

localupdate

Is used to update the system by specifying local rpm files. Only the specified rpm files of which an older version is already installed will be installed, the remaining specified packages will be ignored. If required the enabled repositories will be used to resolve dependencies. Note that the update command will do a local update, if given a filename. This option is maintained for legacy reasons only.

reinstall

Will reinstall the identically versioned package as is currently installed. This does not work for "installonly" packages, like Kernels. reinstall operates on groups, files, provides and filelists just like the "install" command.

downgrade

Will try and downgrade a package from the version currently installed to the previously highest version (or the specified version). The depsolver will not necessarily work, but if you specify all the packages it should work (and thus. all the simple cases will work). Also this does not work

for "installonly" packages, like Kernels. downgrade operates on groups, files, provides, filelists and rpm files just like the "install" command.

deplist

Produces a list of all dependencies and what packages provide those dependencies for the given packages.

repolist

Produces a list of configured repositories. The default is to list all enabled repositories. If you pass -v, for verbose mode, more information is listed. If the first argument is 'enabled', 'disabled' or 'all' then the command will list those types of repos.

You can pass repo id or name arguments, or wildcards which to match against both of those. However if the id or name matches exactly then the repo will be listed even if you are listing enabled repos. and it is disabled.

In non-verbose mode the first column will start with a '*' if the repo. has metalink data and the latest metadata is not local. For non-verbose mode the last column will also display the number of packages in the repo. and (if there are any user specified excludes) the number of packages excluded.

One last special feature of repolist, is that if you are in non-verbose mode then yum will ignore any repo errors and output the information it can get (Eg. "yum clean all; yum -C repolist" will output something, although the package counts/etc. will be zeroed out).

version

Produces a "version" of the rpmbdb, and of the enabled repositories if "all" is given as the first argument. You can also specify version groups in the version-groups configuration file. If you pass -v, for verbose mode, more information is listed. The version is calculated by taking an SHA1 hash of the packages (in sorted order), and the checksum_type/checksum_data entries from the yumdb. Note that this rpmbdb version is now also used significantly within yum (esp. in yum history).

The version command will now show "groups" of packages as a separate version, and so takes sub-commands:

"version grouplist" - List the defined version groups.

"version groupinfo" - Get the complete list of packages within one or more version groups.

"version installed" - This is the default, only show the version information for installed packages.

"version available" - Only show the version information for available packages.

"version all" - Show the version information for installed and available packages.

"version nogroups | nogroups-*" - Just show the main version information.

"version group-*" - Just show the grouped version information, if more arguments are given then only show the data for those groups.

history

The history command allows the user to view what has happened in past transactions (assuming the history_record config. option is set). You can use info/list/packages-list/packages-info/summary to view what happened, undo/redo/rollback to act on that information and new to start a new history file.

The info/list/summary commands take either a transaction id or a package (with wildcards, as in **Specifying package names**), all three can also be passed no arguments. list can be passed the keyword "all" to list all the transactions.

The packages-list/packages-info commands takes a package (with wildcards, as in **Specifying package names**). And show data from the point of view of that package.

The undo/redo/rollback commands take either a single transaction id or the keyword last and an offset from the last transaction (Eg. if you've done 250 transactions, "last" refers to transaction 250, and "last-4" refers to transaction 246).

The undo/redo commands act on the specified transaction, undo'ing or repeating the work of that transaction. While the rollback command will undo all transactions up to the point of the specified transaction. For example, if you have 3 transactions, where package A; B and C were installed respectively. Then "undo 1" will try to remove package A, "redo 1" will try to install package A (if it is not still installed), and "rollback 1" will try to remove packages B and C. Note that after a "rollback 1" you will have a fourth transaction, although the ending rpmdb version (see: yum version) should be the same in transactions 1 and 4.

The addon-info command takes a transaction ID, and the packages-list command takes a package (with wildcards).

The stats command shows some statistics about the current history DB.

The sync commands allows you to change the rpmdb/yumdb data stored for any installed packages, to whatever is in the current rpmdb/yumdb (this is mostly useful when this data was not stored when the package went into the history DB).

In "history list" you can change the behaviour of the 2nd column via. the configuration option history_list_view.

In "history list" output the Altered column also gives some extra information if there was something not good with the transaction (this is also shown at the end of the package column in the packages-list command).

- > - The rpmdb was changed, outside yum, after the transaction.
- < - The rpmdb was changed, outside yum, before the transaction.
- * - The transaction aborted before completion.
- # - The transaction completed, but with a non-zero status.
- E - The transaction completed fine, but had warning/error output during the transaction.
- P - The transaction completed fine, but problems already existed in the rpmdb.
- s - The transaction completed fine, but --skip-broken was enabled and had to skip some packages.

load-transaction

This command will re-load a saved yum transaction file, this allows you to run a transaction on one machine and then use it on another. The two common ways to get a saved yum transaction file are from "yum -q history addon-info last saved_tx" or via. the automatic saves in \$TMPDIR/yum_save_tx.* when a transaction is solved but not run.

check

Checks the local rpmbdb and produces information on any problems it finds. You can pass the check command the arguments "dependencies" or "duplicates", to limit the checking that is performed (the default is "all" which does both).

The info command can also take ranges of transaction ids, of the form start..end, which will then display a merged history as if all the transactions in the range had happened at once. Eg. "history info 1..4" will merge the first four transactions and display them as a single transaction.

help

Produces help, either for all commands or if given a command name then the help for that particular command.

General Options

Most command line options can be set using the configuration file as well and the descriptions indicate the necessary configuration option to set.

-h, --help

Help; display a help message and then quit.

-y, --assumeyes

Assume yes; assume that the answer to any question which would be asked is yes.

Configuration Option: **assumeyes**

-c, --config=[config file]

Specifies the config file location - can take HTTP and FTP URLs and local file paths.

-q, --quiet

Run without output. Note that you likely also want to use -y.

-v, --verbose

Run with a lot of debugging output.

-d, --debuglevel=[number]

Sets the debugging level to [number] - turns up or down the amount of things that are printed.

Practical range: 0 - 10

Configuration Option: **debuglevel**

-e, --errorlevel=[number]

Sets the error level to [number] Practical range 0 - 10. 0 means print only critical errors about which you must be told. 1 means print all errors, even ones that are not overly important. 1+ means print more errors (if any) -e 0 is good for cron jobs.

Configuration Option: **errorlevel**

--rpmverbosity=[name]

Sets the debug level to [name] for rpm scriptlets. 'info' is the default, other options are: 'critical', 'emergency', 'error', 'warn' and 'debug'.

Configuration Option: **rpmverbosity**

-R, --randomwait=[time in minutes]

Sets the maximum amount of time yum will wait before performing a command - it randomizes over the time.

-C, --cacheonly

Tells yum to run entirely from system cache - does not download or update any headers unless it has to to perform the requested action. If you're using this as a user yum will not use the tempcache for the user but will only use the system cache in the system cachedir.

--version

Reports the **yum** version number and installed package versions for everything in history_record_packages (can be added to by plugins).

--showduplicates

Doesn't limit packages to their latest versions in the info, list and search commands (will also affect plugins which use the doPackageLists() API).

--installroot=root

Specifies an alternative installroot, relative to which all packages will be installed.

Configuration Option: **installroot**

--enablerepo=repoidglob

Enables specific repositories by id or glob that have been disabled in the configuration file using the enabled=0 option.

Configuration Option: **enabled**

--disablerepo=repoidglob

Disables specific repositories by id or glob.

Configuration Option: **enabled**

--obsoletes

This option only has affect for an update, it enables **yum**'s obsoletes processing logic. For more information see the **update** command above.

Configuration Option: **obsoletes**

-x, --exclude=package

Exclude a specific package by name or glob from updates on all repositories. Configuration Option: **exclude**

--color=[always|auto|never]

Display colorized output automatically, depending on the output terminal, always (using ANSI codes) or never. Note that some commands (Eg. list and info) will do a little extra work when color is enabled. Configuration Option: **color**

--disableexcludes=[all|main|repoid]

Disable the excludes defined in your config files. Takes one of three options:

all == disable all excludes

main == disable excludes defined in [main] in yum.conf

repoid == disable excludes defined for that repo

--disableplugin=plugin

Run with one or more plugins disabled, the argument is a comma separated list of wildcards to match against plugin names.

--nopugins

Run with all plugins disabled.

Configuration Option: **plugins**

--nogpgcheck

Run with GPG signature checking disabled.

Configuration Option: **gpgcheck**

--skip-broken

Resolve depolve problems by removing packages that are causing problems from the transaction.

Configuration Option: **skip_broken**

--releasever=version

Pretend the current release version is the given string. This is very useful when combined with --installroot. Note that with the default upstream cachedir, of /var/cache/yum, using this option will corrupt your cache (and you can use \$releasever in your cachedir configuration to stop this).

-t, --tolerant

This option currently does nothing.

--setopt=option=value

Set any config option in yum config or repo files. For options in the global config just use: --setopt=option=value for repo options use: --setopt=repoid.option=value

List Options

The following are the ways which you can invoke **yum** in list mode. Note that all **list** commands include information on the version of the package.

OUTPUT

The format of the output of yum list is:

name.arch [epoch:]version-release repo or @installed-from-repo

yum list [all | glob_exp1] [glob_exp2] [...]

List all available and installed packages.

yum list available [glob_exp1] [...]

List all packages in the yum repositories available to be installed.

yum list updates [glob_exp1] [...]

List all packages with updates available in the yum repositories.

yum list installed [glob_exp1] [...]

List the packages specified by *args*. If an argument does not match the name of an available package, it is assumed to be a shell-style glob and any matches are printed.

yum list extras [glob_exp1] [...]

List the packages installed on the system that are not available in any yum repository listed in the config file.

yum list obsoletes [glob_exp1] [...]

List the packages installed on the system that are obsoleted by packages in any yum repository listed in the config file.

yum list recent

List packages recently added into the repositories. This is often not helpful, but what you may really want to use is "yum list-updateinfo new" from the security yum plugin.

Specifying Package Names

A package can be referred to for install, update, remove, list, info etc with any of the following as well as globs of any of the following:

name

name.arch

name-ver
name-ver-rel
name-ver-rel.arch
name-epoch:ver-rel.arch
epoch:name-ver-rel.arch

For example: **yum remove kernel-2.4.1-10.i686**

this will remove this specific kernel-ver-rel.arch.

Or: **yum list available 'foo*'**

will list all available packages that match 'foo*'. (The single quotes will keep your shell from expanding the globs.)

Clean Options

The following are the ways which you can invoke **yum** in clean

mode. Note that "all files" in the commands below means "all files in currently enabled repositories". If you want to also clean any (temporarily) disabled repositories you need to use **-enablerepo='*' option.**

yum clean expire-cache

Eliminate the local data saying when the metadata and mirrorlists were downloaded for each repo. This means yum will revalidate the cache for each repo. next time it is used. However if the cache is still valid, nothing significant was deleted.

yum clean packages

Eliminate any cached packages from the system. Note that packages are not automatically deleted after they are downloaded.

yum clean headers

Eliminate all of the header files, which old versions of yum used for dependency resolution.

yum clean metadata

Eliminate all of the files which yum uses to determine the remote availability of packages. Using this option will force yum to download all the metadata the next time it is run.

yum clean dbcache

Eliminate the sqlite cache used for faster access to metadata. Using this option will force yum to download the sqlite metadata the next time it is run, or recreate the sqlite metadata if using an older repo.

yum clean rpmdb

Eliminate any cached data from the local rpmdb.

yum clean plugins

Tell any enabled plugins to eliminate their cached data.

yum clean all

Does all of the above.

Plugins

Yum can be extended through the use of plugins. A plugin is a Python ".py" file which is installed in one of the directories specified by the **pluginpath** option in yum.conf. For a plugin to work, the following conditions must be met:

1. The plugin module file must be installed in the plugin path as just described.
2. The global **plugins** option in /etc/yum.conf must be set to '1'.

3. A configuration file for the plugin must exist in /etc/yum/pluginconf.d/<plugin_name>.conf and the **enabled** setting in this file must set to '1'. The minimal content for such a configuration file is:

[main]

```
enabled = 1
```

See the **yum.conf(5)** man page for more information on plugin related configuration options.

Files

/etc/yum.conf
/etc/yum/version-groups.conf
/etc/yum.repos.d/
/etc/yum/pluginconf.d/
/var/cache/yum/

See Also

pkcon (1) yum.conf (5) yum-updatesd (8) package-cleanup (1) repoquery (1) yum-complete-transaction (1) yumdownloader (1) yum-utils (1) yum-security (8) <http://yum.baseurl.org/>
<http://yum.baseurl.org/wiki/Faq>
yum search yum

Authors

See the Authors file included with this program.

Bugs

There of course aren't any bugs, but if you find any, you should first consult the FAQ mentioned above and then email the mailing list: yum@lists.baseurl.org or filed in bugzilla.

Referenced By

[febootstrap\(8\)](#), [mock\(1\)](#), [pirut\(8\)](#), [pup\(8\)](#), [puplet\(8\)](#), [system-cdinstall-helper\(8\)](#), [system-install-packages\(8\)](#), [yum-updatesd\(8\)](#), [yumdownloader\(1\)](#)

yum.conf(5) - Linux man page

Name

yum.conf - Configuration file for [yum\(8\)](#).

Description

Yum uses a configuration file at **/etc/yum.conf**.

Additional configuration files are also read from the directories set by the **reposdir** option (default is '/etc/yum.repos.d'). See the **reposdir** option below for further details.

Parameters

There are two types of sections in the yum configuration **file(s)**: main and repository. Main defines all global configuration options. There should be only one main section. The repository **section(s)** define the configuration for each repository/server. There should be one or more repository sections.

[main] OPTIONS

The [main] section must exist for yum to do anything. It consists of the following options:

cachedir Directory where yum should store its cache and db files. The default is '/var/cache/yum'.

persistdir Directory where yum should store information that should persist over multiple runs. The default is '/var/lib/yum'.

keepcache Either '1' or '0'. Determines whether or not yum keeps the cache of headers and packages after successful installation. Default is '1' (keep files)

reposdir A list of directories where yum should look for .repo files which define repositories to use. Default is '/etc/yum.repos.d'. Each file in this directory should contain one or more repository sections as documented in **[repository] options** below. These will be merged with the repositories defined in /etc/yum.conf to form the complete set of repositories that yum will use.

debuglevel Debug message output level. Practical range is 0-10. Default is '2'.

errorlevel Error message output level. Practical range is 0-10. Default is '2'.

rpmverbosity Debug scriptlet output level. 'info' is the default, other options are: 'critical', 'emergency', 'error', 'warn' and 'debug'.

protected_packages This is a list of packages that yum should never completely remove. They are protected via. Obsoletes as well as user/plugin removals.

The default is: yum glob:/etc/yum/protected.d/*.conf So any packages which should be protected can do so by including a file in /etc/yum/protected.d with their package name in it.

Also if this configuration is set to anything, then yum will protect the package corresponding to the running version of the kernel.

protected_multilib Either '1' or '0'. This tells yum whether or not it should perform a check to make sure that multilib packages are the same version. For example, if this option is off (rpm behaviour) pkgA-1.x86_64 and pkgA-2.i386 can be installed at the same time. However this is very rarely desired. install only packages, like the kernel, are exempt from this check. The default is '1'.

logfile Full directory and file name for where yum should write its log file.

gpgcheck Either '1' or '0'. This tells yum whether or not it should perform a GPG signature check on packages. When this is set in the [main] section it sets the default for all repositories. The default is '0'.

localpkg_gpgcheck Either '1' or '0'. This tells yum whether or not it should perform a GPG signature check on local packages (packages in a file, not in a repository). The default is '0'.

repo_gpgcheck Either '1' or '0'. This tells yum whether or not it should perform a GPG signature check on the repodata. When this is set in the [main] section it sets the default for all repositories. The default is '0'.

skip_broken Either '1' or '0'. Resolve dependency problems by removing packages that are causing problems from the transaction.

assumeyes Either '1' or '0'. Determines whether or not yum prompts for confirmation of critical actions. Default is '0' (do prompt).

Command-line option: **-y**

alwaysprompt Either '1' or '0'. Without this option, yum will not prompt for confirmation when the list of packages to be installed exactly matches those given on the command line. Unless **assumeyes** is enabled, it will still prompt for package removal, or when additional packages need to be installed to fulfill dependencies. Default is '1'.

tolerant Either '1' or '0'. If enabled, then yum will be tolerant of errors on the command line with regard to packages. For example: if you request to install foo, bar and baz and baz is installed; yum won't error out complaining that baz is already

installed. Default to '0' (not tolerant).

Command-line option: **-t**

exclude List of packages to exclude from updates or installs. This should be a space separated list. Shell globs using wildcards (eg. * and ?) are allowed.

exactarch Either '1' or '0'. Set to '1' to make yum update only update the architectures of packages that you have installed. ie: with this enabled yum will not install an i686 package to update an i386 package. Default is '1'.

installonlypkgs List of package provides that should only ever be installed, never updated. Kernels in particular fall into this category. Defaults to kernel, kernel-bigmem, kernel-enterprise, kernel-smp, kernel-debug, kernel-unsupported, kernel-source, kernel-devel, kernel-PAE, kernel-PAE-debug.

Note that because these are provides, and not just package names, kernel-devel will also apply to kernel-debug-devel, etc.

Note that "kernel-modules" is **not** in this list, in RHEL-6, and so anything providing that is updated like any other package.

installonly_limit Number of packages listed in installonlypkgs to keep installed at the same time. Setting to 0 disables this feature. Default is '3'. Note that this functionality used to be in the "installonly" plugin, where this option was altered via. tokeep. Note that as of version 3.2.24, yum will now look in the yumdb for a installonly attribute on installed packages. If that attribute is "keep", then they will never be removed.

kernelpkgnames List of package names that are kernels. This is really only here for the updating of kernel packages and should be removed out in the yum 2.1 series.

showduplicatesfromrepos Either '0' or '1'. Set to '1' if you wish to show any duplicate packages from any repository, from package listings like the info or list commands. Set to '0' if you want only to see the newest packages from any repository. Default is '0'.

obsoletes This option only has affect during an **update**. It enables yum's obsoletes processing logic. Useful when doing distribution level upgrades. See also the yum **upgrade** command documentation for more details ([yum\(8\)](#)). Default is 'true'.

Command-line option: **--obsoletes**

overwrite_groups Either '0' or '1'. Used to determine yum's behaviour if two or more repositories offer the package groups with the same name. If **overwrite_groups** is '1' then the group packages of the last matching repository will be used. If **overwrite_groups** is '0' then the groups from all matching repositories will be merged together as one large group.

groupremove_leaf_only Either '0' or '1'. Used to determine yum's behaviour when the groupremove command is run. If **groupremove_leaf_only** is '0' (default) then all packages in the group will be removed. If **groupremove_leaf_only** is '1' then only those packages in the group that aren't required by another package will be removed.

enable_group_conditionals Either '0' or '1'. Determines whether yum will allow the use of conditionals packages. Default is '1' (package conditionals are allowed).

group_package_types List of the following: optional, default, mandatory. Tells yum which type of packages in groups will be installed when 'groupinstall' is called. Default is: default, mandatory

installroot Specifies an alternative installroot, relative to which all packages will be installed.

Command-line option: **--installroot**

distroverpkg The package used by yum to determine the "version" of the distribution. This can be any installed package. Default is 'redhat-release'. You can see what provides this manually by using: "yum whatprovides redhat-release".

diskspacecheck Either '0' or '1'. Set this to '0' to disable the checking for sufficient diskspace before a RPM transaction is run. Default is '1' (perform the check).

tsflags Comma or space separated list of transaction flags to pass to the rpm transaction set. These include 'noscripts', 'notriggers', 'nodocs', 'test', 'justdb' and 'nocontexts'. 'repackage' is also available but that does nothing with newer rpm versions. You can set all/any of them. However, if you don't know what these do in the context of an rpm transaction set you're best leaving it alone. Default is an empty list.

recent Number of days back to look for 'recent' packages added to a repository. Used by the **list recent** command. Default is '7'.

retries Set the number of times any attempt to retrieve a file should retry before returning an error. Setting this to '0' makes yum try forever. Default is '10'.

keepalive Either '0' or '1'. Set whether HTTP keepalive should be used for HTTP/1.1 servers that support it. This can improve transfer speeds by using one connection when downloading multiple files from a repository. Default is '1'.

timeout Number of seconds to wait for a connection before timing out. Defaults to 30 seconds. This may be too short of a time for extremely overloaded sites.

http_caching Determines how upstream HTTP caches are instructed to handle any HTTP downloads that Yum does. This option can take the following values:

'all' means that all HTTP downloads should be cached.

'packages' means that only RPM package downloads should be cached (but not repository metadata downloads).

'none' means that no HTTP downloads should be cached.

The default is 'all'. This is recommended unless you are experiencing caching related issues. Try to at least use 'packages' to minimize load on repository servers.

throttle Enable bandwidth throttling for downloads. This option can be expressed as a absolute data rate in bytes/sec. An SI prefix (k, M or G) may be appended to the bandwidth value (eg. '5.5k' is 5.5 kilobytes/sec, '2M' is 2 Megabytes/sec).

Alternatively, this option can specify the percentage of total bandwidth to use (eg. '60%'). In this case the **bandwidth** option should be used to specify the maximum available bandwidth.

Set to '0' to disable bandwidth throttling. This is the default.

bandwidth Use to specify the maximum available network bandwidth in bytes/second. Used with the **throttle** option (above). If **throttle** is a percentage and **bandwidth** is '0' then bandwidth throttling will be disabled. If **throttle** is expressed as a data rate (bytes/sec) then this option is ignored. Default is '0' (no bandwidth throttling).

sslcacert Path to the directory containing the databases of the certificate authorities yum should use to verify SSL certificates. Defaults to none - uses system default

sslvverify Boolean - should yum verify SSL certificates/hosts at all. Defaults to True.

Note that the plugin yum-rhn-plugin will force this value to true, and may alter other ssl settings (like hostname checking), even if it the machine is not registered.

sslclientcert Path to the SSL client certificate yum should use to connect to repos/remote sites Defaults to none.

Note that if you are using curl compiled against NSS (default in Fedora/RHEL), curl treats sslclientcert values with the same basename as _identical_. This version of yum will check that this isn't true and output an error when the repositories "foo" and "bar" violate this, like so:

sslclientcert basename shared between foo and bar

sslclientkey Path to the SSL client key yum should use to connect to repos/remote sites Defaults to none.

ssl_check_cert_permissions Boolean - Whether yum should check the permissions on the paths for the certificates on the repository (both remote and local). If we can't

read any of the files then yum will force `skip_if_unavailable` to be true. This is most useful for non-root processes which use yum on repos. that have client cert files which are readable only by root. Defaults to True.

history_record Boolean - should yum record history entries for transactions. This takes some disk space, and some extra time in the transactions. But it allows how to know a lot of information about what has happened before, and display it to the user with the history info/list/summary commands. yum also provides the history undo/redo commands. Defaults to True.

Note that if history is recorded, yum uses that information to see if any modifications to the rpmbdb have been done outside of yum. These are always bad, from yum's point of view, and so yum will issue a warning and automatically run some of "yum check" to try and find some of the worst problems altering the rpmbdb might have caused.

This means that turning this option off will stop yum from being able to detect when the rpmbdb has changed and thus. it will never warn you or automatically run "yum check". The problems will likely still be there, and yumdb etc. will still be wrong but yum will not warn you about it.

history_record_packages This is a list of package names that should be recorded as having helped the transaction. yum plugins have an API to add themselves to this, so it should not normally be necessary to add packages here. Note that this is also used for the packages to look for in --version. Defaults to rpm, yum, yum-metadata-parser.

history_list_view Which column of information to display in the "yum history list" command. There are currently three options: users, cmd (or commands), single-user-commands.

Older versions of yum acted like "users", which always outputs the user who initiated the yum transaction. You can now specify "commands" which will instead always output the command line of the transaction. You can also specify "single-user-commands" which will display the users if there are more than one, otherwise it will display the command line.

You can also specify "default" which currently selects "users".

commands List of functional commands to run if no functional commands are specified on the command line (eg. "update foo bar baz quux"). None of the short options (eg. -y, -e, -d) are accepted for this option.

syslog_ident Identification (program name) for syslog messages.

syslog_facility Facility name for syslog messages, see [syslog\(3\)](#). Default is 'LOG_USER'.

syslog_device Where to log syslog messages. Can be a local device (path) or a host:port string to use a remote syslog. If empty or points to a nonexistent device, syslog logging is disabled. Default is '/dev/log'.

proxy URL to the proxy server that yum should use.

proxy_username username to use for proxy

proxy_password password for this proxy

username username to use for basic authentication to a repo or really any url.

password password to use with the username for basic authentication.

plugins Either '0' or '1'. Global switch to enable or disable yum plugins. Default is '0' (plugins disabled). See the **PLUGINS** section of the [yum\(8\)](#) man for more information on installing yum plugins.

pluginpath A list of directories where yum should look for plugin modules. Default is '/usr/share/yum-plugins' and '/usr/lib/yum-plugins'.

pluginconfpath A list of directories where yum should look for plugin configuration files. Default is '/etc/yum/pluginconf.d'.

metadata_expire Time (in seconds) after which the metadata will expire. So that if the current metadata downloaded is less than this many seconds old then yum will not update the metadata against the repository. If you find that yum is not downloading information on updates as often as you would like lower the value of this option. You can also change from the default of using seconds to using days, hours or minutes by appending a d, h or m respectively. The default is 6 hours, to compliment yum-updatesd running once an hour. It's also possible to use the word "never", meaning that the metadata will never expire. Note that when using a metalink file the metalink must always be newer than the metadata for the repository, due to the validation, so this timeout also applies to the metalink file.

mirrorlist_expire Time (in seconds) after which the mirrorlist locally cached will expire. If the current mirrorlist is less than this many seconds old then yum will not download another copy of the mirrorlist, it has the same extra format as metadata_expire. If you find that yum is not downloading the mirrorlists as often as you would like lower the value of this option.

mdpolicy You can select from different metadata download policies depending on how much data you want to download with the main repository metadata index. The advantages of downloading more metadata with the index is that you can't get into situations where you need to use that metadata later and the versions available aren't

compatible (or the user lacks privileges) and that if the metadata is corrupt in any way yum will revert to the previous metadata.

'instant' - Just download the new metadata index, this is roughly what yum always did, however it now does some checking on the index and reverts if it classifies it as bad.

'group:primary' - Download the primary metadata with the index. This contains most of the package information and so is almost always required anyway. This is the default.

'group:small' - With the primary also download the updateinfo metadata, this is required for yum-security operations and it also used in the graphical clients. This file also tends to be significantly smaller than most others.

'group:main' - With the primary and updateinfo download the filelists metadata and the group metadata. The filelists data is required for operations like "yum install /bin/bash", and also some dependency resolutions require it. The group data is used in some graphical clients and for group operations like "yum grouplist Base".

'group:all' - Download all metadata listed in the index, currently the only one not listed above is the other metadata, which contains the changelog information which is used by yum-changelog. This is what "yum makecache" uses.

multilib_policy Can be set to 'all' or 'best'. All means install all possible arches for any package you want to install. Therefore yum install foo will install foo.i386 and foo.x86_64 on x86_64, if it is available. Best means install the best arch for this platform, only.

bugtracker_url URL where bugs should be filed for yum. Configurable for local versions or distro-specific bugtrackers.

color Display colorized output automatically, depending on the output terminal, always (using ANSI codes) or never. Command-line option: **--color**

color_list_installed_older The colorization/highlighting for packages in list/info installed which are older than the latest available package with the same name and arch. Default is 'bold'. Possible values are a comma separated list containing: bold, blink, dim, reverse, underline, fg:black, fg:red, fg:green, fg:yellow, fg:blue, fg:magenta, fg:cyan, fg:white, bg:black, bg:red, bg:green, bg:yellow, bg:blue, bg:magenta, bg:cyan, bg:white.

color_list_installed_newer The colorization/highlighting for packages in list/info installed which are newer than the latest available package with the same name and arch. Default is 'bold,yellow'. See **color_list_installed_older** for possible values.

color_list_installed_reinstall The colorization/highlighting for packages in list/info installed which is the same version as the latest available package with the same name and arch. Default is 'normal'. See color_list_installed_older for possible values.

color_list_installed_extra The colorization/highlighting for packages in list/info installed which has no available package with the same name and arch. Default is 'bold,red'. See color_list_installed_older for possible values.

color_list_available_upgrade The colorization/highlighting for packages in list/info available which is an upgrade for the latest installed package with the same name and arch. Default is 'bold,blue'. See color_list_installed_older for possible values.

color_list_available_downgrade The colorization/highlighting for packages in list/info available which is a downgrade for the latest installed package with the same name and arch. Default is 'dim,cyan'. See color_list_installed_older for possible values.

color_list_available_install The colorization/highlighting for packages in list/info available which has no installed package with the same name and arch. Default is 'normal'. See color_list_installed_older for possible values.

color_list_available_reinstall The colorization/highlighting for packages in list/info available which is the same version as the installed package with the same name and arch. Default is 'bold,underline,green'. See color_list_installed_older for possible values.

color_search_match The colorization/highlighting for text matches in search. Default is 'bold'. See color_list_installed_older for possible values.

color_update_installed The colorization/highlighting for packages in the "updates list" which are installed. The updates list is what is printed when you run "yum update", "yum list updates", "yum list obsoletes" and "yum check-update". Default is 'normal'. See color_list_installed_older for possible values.

color_update_local The colorization/highlighting for packages in the "updates list" which are already downloaded. The updates list is what is printed when you run "yum update", "yum list updates", "yum list obsoletes" and "yum check-update". Default is 'bold'. See color_list_installed_older for possible values.

color_update_remote The colorization/highlighting for packages in the "updates list" which need to be downloaded. The updates list is what is printed when you run "yum update", "yum list updates", "yum list obsoletes" and "yum check-update". Default is 'normal'. See color_list_installed_older for possible values.

clean_requirements_on_remove When removing packages (by removal, update or obsolescence) go through each package's dependencies. If any of them are no longer

required by any other package then also mark them to be removed. Boolean (1, 0, True, False, yes,no) Defaults to False

reset_nice If set to true then yum will try to reset the nice value to zero, before running an rpm transaction. Defaults to True.

[repository] OPTIONS

The repository **section**(s) take the following form:

Example: [repositoryid]

name=Some name for this repository

baseurl=url://path/to/repository/

repositoryid Must be a unique name for each repository, one word.

name A human readable string describing the repository.

baseurl Must be a URL to the directory where the yum repository's 'repodata' directory lives. Can be an http://, ftp:// or file:// URL. You can specify multiple URLs in one baseurl statement. The best way to do this is like this:

[repositoryid]

name=Some name for this repository

baseurl=url://server1/path/to/repository/

url://server2/path/to/repository/

url://server3/path/to/repository/

If you list more than one baseurl= statement in a repository you will find yum will ignore the earlier ones and probably act bizarrely. Don't do this, you've been warned.

You can use HTTP basic auth by prepending "user:password@" to the server name in the baseurl line. For example: "baseurl=http://user:passwd@example.com/".

metalink Specifies a URL to a metalink file for the repomd.xml, a list of mirrors for the entire repository are generated by converting the mirrors for the repomd.xml file to a baseurl. The metalink file also contains the latest timestamp from the data in the repomd.xml, the length of the repomd.xml and checksum data. This data is checked against any downloaded repomd.xml file and all of the information from the metalink file must match. This can be used instead of or with the **baseurl** option. Substitution variables, described below, can be used with this option. This option disables the mirrorlist option. As a special hack if the mirrorlist URL contains the word "metalink" then the value of mirrorlist is copied to metalink (if metalink is not set).

mirrorlist Specifies a URL to a file containing a list of baseurls. This can be used instead of or with the **baseurl** option. Substitution variables, described below, can be

used with this option. As a special hack is the mirrorlist URL contains the word "metalink" then the value of mirrorlist is copied to metalink (if metalink is not set).

enabled Either '1' or '0'. This tells yum whether or not use this repository.

gpgcheck Either '1' or '0'. This tells yum whether or not it should perform a GPG signature check on the packages gotten from this repository.

repo_gpgcheck Either '1' or '0'. This tells yum whether or not it should perform a GPG signature check on the repodata from this repository.

gpgkey A URL pointing to the ASCII-armored GPG key file for the repository. This option is used if yum needs a public key to verify a package and the required key hasn't been imported into the RPM database. If this option is set, yum will automatically import the key from the specified URL. You will be prompted before the key is installed unless the **assumeyes** option is set.

Multiple URLs may be specified here in the same manner as the **baseurl** option (above). If a GPG key is required to install a package from a repository, all keys specified for that repository will be installed.

gpgcakey A URL pointing to the ASCII-armored CA key file for the repository. This is a normal gpg public key - but this key will be used to validate detached signatures of all other keys. The idea is you are asked to confirm import for this key. After that any other gpg key needed for package or repository verification, if it has a detached signature which matches this key will be automatically imported without user confirmation.

exclude Same as the [main] **exclude** option but only for this repository. Substitution variables, described below, are honored here.

includepkgs Inverse of exclude. This is a list of packages you want to use from a repository. If this option lists only one package then that is all yum will ever see from the repository. Defaults to an empty list. Substitution variables, described below, are honored here.

enablegroups Either '0' or '1'. Determines whether yum will allow the use of package groups for this repository. Default is '1' (package groups are allowed).

failovermethod Either 'roundrobin' or 'priority'.

'roundrobin' randomly selects a URL out of the list of URLs to start with and proceeds through each of them as it encounters a failure contacting the host.

'priority' starts from the first baseurl listed and reads through them sequentially.

failovermethod defaults to 'roundrobin' if not specified.

keepalive Either '1' or '0'. This tells yum whether or not HTTP/1.1 keepalive should be used with this repository. See the global option in the [main] section above for more information.

timeout Overrides the **timeout** option from the [main] section for this repository.

http_caching Overrides the **http_caching** option from the [main] section for this repository.

retries Overrides the **retries** option from the [main] section for this repository.

throttle Overrides the **throttle** option from the [main] section for this repository.

bandwidth Overrides the **bandwidth** option from the [main] section for this repository.

sslcacert Overrides the **sslcacert** option from the [main] section for this repository.

sslverify Overrides the **sslverify** option from the [main] section for this repository.

sslclientcert Overrides the **sslclientcert** option from the [main] section for this repository.

sslclientkey Overrides the **sslclientkey** option from the [main] section for this repository.

ssl_check_cert_permissions Overrides the **ssl_check_cert_permissions** option from the [main] section for this repository.

metadata_expire Overrides the **metadata_expire** option from the [main] section for this repository.

mirrorlist_expire Overrides the **mirrorlist_expire** option from the [main] section for this repository.

proxy URL to the proxy server for this repository. Set to '_none_' to disable the global proxy setting for this repository. If this is unset it inherits it from the global setting

proxy_username username to use for proxy. If this is unset it inherits it from the global setting

proxy_password password for this proxy. If this is unset it inherits it from the global setting

username username to use for basic authentication to a repo or really any url. If this is unset it inherits it from the global setting

password password to use with the username for basic authentication. If this is unset it inherits it from the global setting

cost relative cost of accessing this repository. Useful for weighing one repo's packages as greater/less than any other. defaults to 1000

skip_if_unavailable If set to True yum will continue running if this repository cannot be contacted for any reason. This should be set carefully as all repos are consulted for any given command. Defaults to False.

Url Include Syntax

The inclusion of external configuration files is supported for /etc/yum.conf and the .repo files in the /etc/yum.repos.d directory. To include a URL, use a line of the following format:

```
include=url://to/some/location
```

The configuration file will be inserted at the position of the "include=" line. Included files may contain further include lines. Yum will abort with an error if an inclusion loop is detected.

GLOB: FOR LIST OPTIONS

Any of the configurations options which are a list of items can be specified using the glob syntax: **glob:/etc/path/somewhere.d/*.conf**. This will read in all files matching that glob and include all lines in each file (excluding comments and blank lines) as items in the list.

Variables

There are a number of variables you can use to ease maintenance of yum's configuration files. They are available in the values of several options including **name**, **baseurl** and **commands**.

\$releasever This will be replaced with the value of the version of the package listed in **distroverpkg**. This defaults to the version of 'redhat-release' package.

\$arch This will be replaced with your architecture as listed by os.uname()[4] in Python.

\$basearch This will be replaced with your base architecture in yum. For example, if your \$arch is i686 your \$basearch will be i386.

\$uuid This will be replaced with a unique but persistent uuid for this machine. The value that is first generated will be stored in /var/lib/yum/uuid and reused until this file is deleted.

\$YUM0-\$YUM9 These will be replaced with the value of the shell environment variable of the same name. If the shell environment variable does not exist then the configuration file variable will not be replaced.

As of 3.2.28, any file in /etc/yum/vars is turned into a variable named after the filename (or overrides any of the above variables).

Note that no warnings/errors are given if the files are unreadable, so creating files that only root can read may be confusing for users.

Also note that only the first line will be read and all new line characters are removed, as a convenience. However, no other checking is performed on the data. This means it is possible to have bad character data in any value.

Files

/etc/yum.conf
/etc/yum.repos.d/
/etc/yum/pluginconf.d/
/etc/yum/protected.d
/etc/yum/vars

See Also

[yum\(8\)](#)

Referenced By

[febootstrap3\(8\)](#), [yum-updatesd.conf\(5\)](#)

NAME

`zsh` – the Z shell

OVERVIEW

Because `zsh` contains many features, the `zsh` manual has been split into a number of sections:

`zsh` Zsh overview (this section)
`zshroadmap` Informal introduction to the manual
`zshmisc` Anything not fitting into the other sections
`zshexpn` Zsh command and parameter expansion
`zshparam` Zsh parameters
`zshoptions` Zsh options
`zshbuiltins` Zsh built-in functions
`zshzle` Zsh command line editing
`zshcompwid` Zsh completion widgets
`zshcompsys` Zsh completion system
`zshcompctl` Zsh completion control
`zshmodules` Zsh loadable modules
`zshcalsys` Zsh built-in calendar functions
`zshtcp sys` Zsh built-in TCP functions
`zshzftpsys` Zsh built-in FTP client
`zshcontrib` Additional zsh functions and utilities
`zshall` Meta-man page containing all of the above

DESCRIPTION

Zsh is a UNIX command interpreter (shell) usable as an interactive login shell and as a shell script command processor. Of the standard shells, `zsh` most closely resembles `ksh` but includes many enhancements. It does not provide compatibility with POSIX or other shells in its default operating mode: see the section Compatibility below.

Zsh has command line editing, builtin spelling correction, programmable command completion, shell functions (with autoloading), a history mechanism, and a host of other features.

AUTHOR

Zsh was originally written by Paul Falstad <pf@zsh.org>. Zsh is now maintained by the members of the zsh-workers mailing list <zsh-workers@zsh.org>. The development is currently coordinated by Peter Stephenson <pws@zsh.org>. The coordinator can be contacted at<coordinator@zsh.org>, but matters relating to the code should generally go to the mailing list.

AVAILABILITY

Zsh is available from the following HTTP and anonymous FTP site.

<ftp://ftp.zsh.org/pub/>
<https://www.zsh.org/pub/>
)

The up-to-date source code is available via Git from Sourceforge. See<https://sourceforge.net/projects/zsh/> for details. A summary of instructions for the archive can be found at <http://zsh.sourceforge.net/>.

MAILING LISTS

Zsh has 3 mailing lists:

<zsh-announce@zsh.org>

Announcements about releases, major changes in the shell and the monthly posting of the Zsh FAQ. (moderated)

<zsh-users@zsh.org>

User discussions.

<zsh-workers@zsh.org>

Hacking, development, bug reports and patches.

To subscribe or unsubscribe, send mail to the associated administrative address for the mailing list.

<zsh-announce-subscribe@zsh.org>

<zsh-users-subscribe@zsh.org>

<zsh-workers-subscribe@zsh.org>

<zsh-announce-unsubscribe@zsh.org>

<zsh-users-unsubscribe@zsh.org>

<zsh-workers-unsubscribe@zsh.org>

YOU ONLY NEED TO JOIN ONE OF THE MAILING LISTS AS THEY ARE NESTED. All submissions to **zsh-announce** are automatically forwarded to **zsh-users**. All submissions to **zsh-users** are automatically forwarded to **zsh-workers**.

If you have problems subscribing/unsubscribing to any of the mailing lists, send mail to **<listmaster@zsh.org>**. The mailing lists are maintained by Karsten Thygesen **<kathy@kom.auc.dk>**.

The mailing lists are archived; the archives can be accessed via the administrative addresses listed above. There is also a hypertext archive, maintained by Geoff Wing **<gew@zsh.org>**, available at <https://www.zsh.org/mla/>.

THE ZSH FAQ

Zsh has a list of Frequently Asked Questions (FAQ), maintained by Peter Stephenson **<pws@zsh.org>**. It is regularly posted to the newsgroup **comp.unix.shell** and the **zsh-announce** mailing list. The latest version can be found at any of the Zsh FTP sites, or at <http://www.zsh.org/FAQ/>. The contact address for FAQ-related matters is **<faqmaster@zsh.org>**.

THE ZSH WEB PAGE

Zsh has a web page which is located at <https://www.zsh.org/>. This is maintained by Karsten Thygesen **<kathy@zsh.org>**, of SunSITE Denmark. The contact address for web-related matters is **<webmaster@zsh.org>**.

THE ZSH USERGUIDE

A userguide is currently in preparation. It is intended to complement the manual, with explanations and hints on issues where the manual can be cabballistic, hierographic, or downright mystifying (for example, the word ‘hierographic’ does not exist). It can be viewed in its current state at <http://zsh.sourceforge.net/Guide/>. At the time of writing, chapters dealing with startup files and their contents and the new completion system were essentially complete.

INVOCATION

The following flags are interpreted by the shell when invoked to determine where the shell will read commands from:

- c Take the first argument as a command to execute, rather than reading commands from a script or standard input. If any further arguments are given, the first one is assigned to \$0, rather than being used as a positional parameter.
- i Force shell to be interactive. It is still possible to specify a script to execute.
- s Force shell to read commands from the standard input. If the -s flag is not present and an argument is given, the first argument is taken to be the pathname of a script to execute.

If there are any remaining arguments after option processing, and neither of the options -c or -s was supplied, the first argument is taken as the file name of a script containing shell commands to be executed. If the option **PATH_SCRIPT** is set, and the file name does not contain a directory path (i.e. there is no ‘/’ in the name), first the current directory and then the command path given by the variable **PATH** are searched for the script. If the option is not set or the file name contains a ‘/’ it is used directly.

After the first one or two arguments have been appropriated as described above, the remaining arguments are assigned to the positional parameters.

For further options, which are common to invocation and the **set** builtin, see *zshoptions(1)*.

The long option ‘**--emulate**’ followed (in a separate word) by an emulation mode may be passed to the shell. The emulation modes are those described for the **emulate** builtin, see *zshbuiltins(1)*. The ‘**--emulate**’ option must precede any other options (which might otherwise be overridden), but following options are honoured, so may be used to modify the requested emulation mode. Note that certain extra steps are taken to ensure a smooth emulation when this option is used compared with the **emulate** command within the shell: for example, variables that conflict with POSIX usage such as **path** are not defined within the shell.

Options may be specified by name using the **-o** option. **-o** acts like a single-letter option, but takes a following string as the option name. For example,

```
zsh -x -o shwordsplit scr
```

runs the script **scr**, setting the **XTRACE** option by the corresponding letter ‘**-x**’ and the **SH_WORD_SPLIT** option by name. Options may be turned *off* by name by using **+o** instead of **-o**. **-o** can be stacked up with preceding single-letter options, so for example ‘**-xo shwordsplit**’ or ‘**-xoshword-split**’ is equivalent to ‘**-x -o shwordsplit**’.

Options may also be specified by name in GNU long option style, ‘**--option-name**’. When this is done, ‘**-**’ characters in the option name are permitted: they are translated into ‘**_**’, and thus ignored. So, for example, ‘**zsh --sh-word-split**’ invokes zsh with the **SH_WORD_SPLIT** option turned on. Like other option syntaxes, options can be turned off by replacing the initial ‘**-**’ with a ‘**+**’; thus ‘**+sh-word-split**’ is equivalent to ‘**--no-sh-word-split**’. Unlike other option syntaxes, GNU-style long options cannot be stacked with any other options, so for example ‘**-x-shwordsplit**’ is an error, rather than being treated like ‘**-x --shwordsplit**’.

The special GNU-style option ‘**--version**’ is handled; it sends to standard output the shell’s version information, then exits successfully. ‘**--help**’ is also handled; it sends to standard output a list of options that can be used when invoking the shell, then exits successfully.

Option processing may be finished, allowing following arguments that start with ‘**-**’ or ‘**+**’ to be treated as normal arguments, in two ways. Firstly, a lone ‘**-**’ (or ‘**+**’) as an argument by itself ends option processing. Secondly, a special option ‘**--**’ (or ‘**+--**’), which may be specified on its own (which is the standard POSIX usage) or may be stacked with preceding options (so ‘**-x--**’ is equivalent to ‘**-x --**’). Options are not permitted to be stacked after ‘**--**’ (so ‘**-x-f**’ is an error), but note the GNU-style option form discussed above, where ‘**--shwordsplit**’ is permitted and does not end option processing.

Except when the **sh/ksh** emulation single-letter options are in effect, the option ‘**-b**’ (or ‘**+b**’) ends option processing. ‘**-b**’ is like ‘**--**’, except that further single-letter options can be stacked after the ‘**-b**’ and will take effect as normal.

COMPATIBILITY

Zsh tries to emulate **sh** or **ksh** when it is invoked as **sh** or **ksh** respectively; more precisely, it looks at the first letter of the name by which it was invoked, excluding any initial ‘**r**’ (assumed to stand for ‘restricted’), and if that is ‘**b**’, ‘**s**’ or ‘**k**’ it will emulate **sh** or **ksh**. Furthermore, if invoked as **su** (which happens on certain systems when the shell is executed by the **su** command), the shell will try to find an alternative name from the **SHELL** environment variable and perform emulation based on that.

In **sh** and **ksh** compatibility modes the following parameters are not special and not initialized by the shell: **ARGC**, **argv**, **cdpath**, **fignore**, **fpath**, **HISTCHARS**, **mailpath**, **MANPATH**, **manpath**, **path**, **prompt**, **PROMPT**, **PROMPT2**, **PROMPT3**, **PROMPT4**, **psvar**, **status**, **watch**.

The usual zsh startup/shutdown scripts are not executed. Login shells source **/etc/pr file** followed by **\$HOME/.profile**. If the **ENV** environment variable is set on invocation, **\$ENV** is sourced after the profile scripts. The value of **ENV** is subjected to parameter expansion, command substitution, and arithmetic expansion before being interpreted as a pathname. Note that the **PRIVILEGED** option also affects the execution of startup files.

The following options are set if the shell is invoked as **sh** or **ksh**: **NO_BAD_PATTERN**, **NO_BANG_HIST**, **NO_BG_NICE**, **NO_EQUALS**, **NO_FUNCTION_ARGZERO**, **GLOB_SUBST**,

NO_GLOBAL_EXPORT, **NO_HUP**, **INTERACTIVE_COMMENTS**, **KSH_ARRAYS**, **NO_MULTIOS**, **NO_NOMATCH**, **NO_NOTIFY**, **POSIX_BUILTINS**, **NO_PROMPT_PERCENT**, **RM_STAR_SILENT**, **SH_FILE_EXPANSION**, **SH_GLOB**, **SH_OPTION LETTERS**, **SH_WORD_SPLIT**. Additionally the **BSD_ECHO** and **IGNORE_BRA CES** options are set if zsh is invoked as **sh**. Also, the **KSH_OPTION_PRINT**, **LOCAL_OPTIONS**, **PR OMPT_BANG**, **PROMPT_SUBST** and **SINGLE_LINE_ZLE** options are set if zsh is invoked as **ksh**.

RESTRICTED SHELL

When the basename of the command used to invoke zsh starts with the letter ‘r’ or the ‘-r’ command line option is supplied at invocation, the shell becomes restricted. Emulation mode is determined after stripping the letter ‘r’ from the invocation name. The following are disabled in restricted mode:

- changing directories with the **cd** builtin
- changing or unsetting the **EGID**, **EUID**, **GID**, **HISTFILE**, **HISTSIZE**, **IFS**, **LD_AOUT_LIBRARY_PATH**, **LD_AOUT_PRELOAD**, **LD_LIBRARY_PATH**, **LD_PRELOAD**, **MODULE_PATH**, **module_path**, **PATH**, **path**, **SHELL**, **UID** and **USERNAME** parameters
- specifying command names containing /
- specifying command pathnames using **hash**
- redirecting output to files
- using the **exec** builtin command to replace the shell with another command
- using **jobs -Z** to overwrite the shell process’ argument and environment space
- using the **ARGV0** parameter to override **argv[0]** for external commands
- turning off restricted mode with **set +r** or **unsetopt RESTRICTED**

These restrictions are enforced after processing the startup files. The startup files should set up **PATH** to point to a directory of commands which can be safely invoked in the restricted environment. They may also add further restrictions by disabling selected builtins.

Restricted mode can also be activated any time by setting the **RESTRICTED** option. This immediately enables all the restrictions described above even if the shell still has not processed all startup files.

A shell *Restricted Mode* is an outdated way to restrict what users may do: modern systems have better, safer and more reliable ways to confine user actions, such as *chroot jails*, *containers* and *zones*.

A restricted shell is very difficult to implement safely. The feature may be removed in a future version of zsh.

It is important to realise that the restrictions only apply to the shell, not to the commands it runs (except for some shell builtins). While a restricted shell can only run the restricted list of commands accessible via the predefined ‘**PATH**’ variable, it does not prevent those commands from running any other command.

As an example, if ‘**env**’ is among the list of *allowed* commands, then it allows the user to run any command as ‘**env**’ is not a shell builtin command and can run arbitrary executables.

So when implementing a restricted shell framework it is important to be fully aware of what actions each of the *allowed* commands or features (which may be regarded as *modules*) can perform.

Many commands can have their behaviour affected by environment variables. Except for the few listed above, zsh does not restrict the setting of environment variables.

If a ‘**perl**’, ‘**python**’, ‘**bash**’, or other general purpose interpreted script is treated as a restricted command, the user can work around the restriction by setting specially crafted ‘**PERL5LIB**’, ‘**PYTHONPATH**’, ‘**BASHENV**’ (etc.) environment variables. On GNU systems, any command can be made to run arbitrary code when performing character set conversion (including zsh itself) by setting a ‘**GCONV_PATH**’ environment variable. Those are only a few examples.

Bear in mind that, contrary to some other shells, ‘**readonly**’ is not a security feature in zsh as it can be undone and so cannot be used to mitigate the above.

A restricted shell only works if the allowed commands are few and carefully written so as not to grant more access to users than intended. It is also important to restrict what zsh module the user may load as some of them, such as ‘zsh/system’, ‘zsh/mapfile’ and ‘zsh/files’, allow bypassing most of the restrictions.

STARTUP/SHUTDOWN FILES

Commands are first read from `/etc/zsh/zshenv`; this cannot be overridden. Subsequent behaviour is modified by the **RCS** and **GLOBAL_RCS** options; the former affects all startup files, while the second only affects global startup files (those shown here with a `/`). If one of the options is unset at any point, any subsequent startup file(s) of the corresponding type will not be read. It is also possible for a file in `$ZDOTDIR` to re-enable **GLOBAL_RCS**. Both **RCS** and **GLOBAL_RCS** are set by default.

Commands are then read from `$ZDOTDIR/.zshenv`. If the shell is a login shell, commands are read from `/etc/zsh/zprofile` and then `$ZDOTDIR/.zprofile`. Then, if the shell is interactive, commands are read from `/etc/zsh/zshrc` and then `$ZDOTDIR/.zshrc`. Finally, if the shell is a login shell, `/etc/zsh/zlogin` and `$ZDOTDIR/.zlogin` are read.

When a login shell exits, the files `$ZDOTDIR/.zlogout` and then `/etc/zsh/zlogout` are read. This happens with either an explicit exit via the **exit** or **logout** commands, or an implicit exit by reading end-of-file from the terminal. However, if the shell terminates due to **exec**'ing another process, the logout files are not read. These are also affected by the **RCS** and **GLOBAL_RCS** options. Note also that the **RCS** option affects the saving of history files, i.e. if **RCS** is unset when the shell exits, no history file will be saved.

If **ZDOTDIR** is unset, **HOME** is used instead. Files listed above as being in `/etc` may be in another directory, depending on the installation.

As `/etc/zsh/zshenv` is run for all instances of zsh, it is important that it be kept as small as possible. In particular, it is a good idea to put code that does not need to be run for every single shell behind a test of the form ‘**if [[-o rcs]]; then ...**’ so that it will not be executed when zsh is invoked with the ‘**-f**’ option.

Any of these files may be pre-compiled with the **zcompile** builtin command (see *zshbuiltins(1)*). If a compiled file exists (named for the original file plus the `.zwc` extension) and it is newer than the original file, the compiled file will be used instead.

FILES

```
$ZDOTDIR/.zshenv
$ZDOTDIR/.zprofile
$ZDOTDIR/.zshrc
$ZDOTDIR/.zlogin
$ZDOTDIR/.zlogout
${TMPPREFIX}* (default is /tmp/zsh*)
/etc/zsh/zshenv
/etc/zsh/zprofile
/etc/zsh/zshrc
/etc/zsh/zlogin
/etc/zsh/zlogout (installation-specific – /etc is the default)
```

SEE ALSO

sh(1), *csh(1)*, *tcsh(1)*, *rc(1)*, *bash(1)*, *ksh(1)*, *zshall(1)*, *zshbuiltins(1)*, *zshcalsys(1)*, *zshcompwid(1)*, *zshcompsys(1)*, *zshcomptcl(1)*, *zshcontrib(1)*, *zshexpn(1)*, *zshmisc(1)*, *zshmodules(1)*, *zshoptions(1)*, *zshparam(1)*, *zshroadmap(1)*, *zshtcpsys(1)*, *zshzftpsys(1)*, *zshzle(1)*

IEEE Standard for information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and Utilities, IEEE Inc, 1993, ISBN 1-55937-255-9.

NAME

`zshbuiltins` – zsh built-in commands

SHELL BUILTIN COMMANDS

Some shell builtin commands take options as described in individual entries; these are often referred to in the list below as ‘**flags**’ to avoid confusion with shell options, which may also have an effect on the behaviour of builtin commands. In this introductory section, ‘**option**’ always has the meaning of an option to a command that should be familiar to most command line users.

Typically, options are single letters preceded by a hyphen (–). Options that take an argument accept it either immediately following the option letter or after white space, for example ‘**print –C3 {1..9}**’ or ‘**print –C 3 {1..9}**’ are equivalent. Arguments to options are not the same as arguments to the command; the documentation indicates which is which. Options that do not take an argument may be combined in a single word, for example ‘**print –rca -- ***’ and ‘**print –r –c –a -- ***’ are equivalent.

Some shell builtin commands also take options that begin with ‘+’ instead of ‘–’. The list below makes clear which commands these are.

Options (together with their individual arguments, if any) must appear in a group before any non-option arguments; once the first non-option argument has been found, option processing is terminated.

All builtin commands other than ‘**echo**’ and precommand modifiers, even those that have no options, can be given the argument ‘**--**’ to terminate option processing. This indicates that the following words are non-option arguments, but is otherwise ignored. This is useful in cases where arguments to the command may begin with ‘–’. For historical reasons, most builtin commands (including ‘**echo**’) also recognize a single ‘–’ in a separate word for this purpose; note that this is less standard and use of ‘**--**’ is recommended.

– simple command

See the section ‘Precommand Modifiers’ in *zshmisc(1)*.

.file [arg ...]

Read commands from *file* and execute them in the current shell environment.

If *file* does not contain a slash, or if **PATH_DIRS** is set, the shell looks in the components of **\$path** to find the directory containing *file*. Files in the current directory are not read unless ‘.’ appears somewhere in **\$path**. If a file named ‘*file.zwc*’ is found, is newer than *file*, and is the compiled form (created with the **zcompile** builtin) of *file*, then commands are read from that file instead of *file*.

If any arguments *arg* are given, they become the positional parameters; the old positional parameters are restored when the *file* is done executing. However, if no arguments are given, the positional parameters remain those of the calling context, and no restoring is done.

If *file* was not found the return status is 127; if *file* was found but contained a syntax error the return status is 126; else the return status is the exit status of the last command executed.

: [arg ...]

This command does nothing, although normal argument expansions is performed which may have effects on shell parameters. A zero exit status is returned.

alias [{+|-}gmrsL] [name[=value] ...]

For each *name* with a corresponding *value*, define an alias with that value. A trailing space in *value* causes the next word to be checked for alias expansion. If the **-g** flag is present, define a global alias; global aliases are expanded even if they do not occur in command position.

If the **-s** flag is present, define a suffix alias: if the command word on a command line is in the form ‘*text.name*’, where *text* is any non-empty string, it is replaced by the text ‘*value text.name*’. Note that *name* is treated as a literal string, not a pattern. A trailing space in *value* is not special in this case. For example,

```
alias -s ps='gv --'
```

will cause the command ‘*.ps’ to be expanded to ‘gv -- *.ps’. As alias expansion is carried out

earlier than globbing, the ‘*.ps’ will then be expanded. Suffix aliases constitute a different name space from other aliases (so in the above example it is still possible to create an alias for the command **ps**) and the two sets are never listed together.

For each *name* with no *value*, print the value of *name*, if any. With no arguments, print all currently defined aliases other than suffix aliases. If the **-m** flag is given the arguments are taken as patterns (they should be quoted to preserve them from being interpreted as glob patterns), and the aliases matching these patterns are printed. When printing aliases and one of the **-g**, **-r** or **-s** flags is present, restrict the printing to global, regular or suffix aliases, respectively; a regular alias is one which is neither a global nor a suffix alias. Using ‘+’ instead of ‘-’, or ending the option list with a single ‘+’, prevents the values of the aliases from being printed.

If the **-L** flag is present, then print each alias in a manner suitable for putting in a startup script. The exit status is nonzero if a *name* (with no *value*) is given for which no alias has been defined.

For more on aliases, include common problems, see the section ALIASING in *zshmisc(1)*.

autoload [{+|-}RTUXdkmrtWz] [-w] [*name* ...]

See the section ‘Autoloading Functions’ in *zshmisc(1)* for full details. The **fpath** parameter will be searched to find the function definition when the function is first referenced.

If *name* consists of an absolute path, the function is defined to load from the file given (searching as usual for dump files in the given location). The name of the function is the basename (non-directory part) of the file. It is normally an error if the function is not found in the given location; however, if the option **-d** is given, searching for the function defaults to **\$fpath**. If a function is loaded by absolute path, any functions loaded from it that are marked for **autoload** without an absolute path have the load path of the parent function temporarily prepended to **\$fpath**.

If the option **-r** or **-R** is given, the function is searched for immediately and the location is recorded internally for use when the function is executed; a relative path is expanded using the value of **\$PWD**. This protects against a change to **\$fpath** after the call to **autoload**. With **-r**, if the function is not found, it is silently left unresolved until execution; with **-R**, an error message is printed and command processing aborted immediately the search fails, i.e. at the **autoload** command rather than at function execution..

The flag **-X** may be used only inside a shell function. It causes the calling function to be marked for autoloading and then immediately loaded and executed, with the current array of positional parameters as arguments. This replaces the previous definition of the function. If no function definition is found, an error is printed and the function remains undefined and marked for autoloading. If an argument is given, it is used as a directory (i.e. it does not include the name of the function) in which the function is to be found; this may be combined with the **-d** option to allow the function search to default to **\$fpath** if it is not in the given location.

The flag **+X** attempts to load each *name* as an autoloaded function, but does *not* execute it. The exit status is zero (success) if the function was not previously defined *and* a definition for it was found. This *does not* replace any existing definition of the function. The exit status is nonzero (failure) if the function was already defined or when no definition was found. In the latter case the function remains undefined and marked for autoloading. If ksh-style autoloading is enabled, the function created will contain the contents of the file plus a call to the function itself appended to it, thus giving normal ksh autoloading behaviour on the first call to the function. If the **-m** flag is also given each *name* is treated as a pattern and all functions already marked for autoload that match the pattern are loaded.

With the **-t** flag, turn on execution tracing; with **-T**, turn on execution tracing only for the current function, turning it off on entry to any called functions that do not also have tracing enabled.

With the **-U** flag, alias expansion is suppressed when the function is loaded.

With the **-w** flag, the *names* are taken as names of files compiled with the **zcompile** builtin, and all functions defined in them are marked for autoloading.

The flags **-z** and **-k** mark the function to be autoloaded using the zsh or ksh style, as if the option

KSH_AUTOLOAD were unset or were set, respectively. The flags override the setting of the option at the time the function is loaded.

Note that the **autoload** command makes no attempt to ensure the shell options set during the loading or execution of the file have any particular value. For this, the **emulate** command can be used:

```
emulate zsh -c 'autoload -Uz func'
```

arranges that when *func* is loaded the shell is in native **zsh** emulation, and this emulation is also applied when *func* is run.

Some of the functions of **autoload** are also provided by **functions -u** or **functions -U**, but **autoload** is a more comprehensive interface.

bg [*job* ...]

job ... &

Put each specified *job* in the background, or the current job if none is specified.

bindkey

See the section ‘Zle Builtins’ in *zshzle(1)*.

break [*n*]

Exit from an enclosing **for**, **while**, **until**, **select** or **repeat** loop. If an arithmetic expression *n* is specified, then break *n* levels instead of just one.

builtin *name* [*args* ...]

Executes the builtin *name*, with the given *args*.

bye Same as **exit**.

cap See the section ‘The zsh/cap Module’ in *zshmodules(1)*.

cd [**-qsLP**] [*arg*]

cd [**-qsLP**] *old new*

cd [**-qsLP**] {+|-}n

Change the current directory. In the first form, change the current directory to *arg*, or to the value of **\$HOME** if *arg* is not specified. If *arg* is ‘-’, change to the previous directory.

Otherwise, if *arg* begins with a slash, attempt to change to the directory given by *arg*.

If *arg* does not begin with a slash, the behaviour depends on whether the current directory ‘.’ occurs in the list of directories contained in the shell parameter **cdpath**. If it does not, first attempt to change to the directory *arg* under the current directory, and if that fails but **cdpath** is set and contains at least one element attempt to change to the directory *arg* under each component of **cdpath** in turn until successful. If ‘.’ occurs in **cdpath**, then **cdpath** is searched strictly in order so that ‘.’ is only tried at the appropriate point.

The order of testing **cdpath** is modified if the option **POSIX_CD** is set, as described in the documentation for the option.

If no directory is found, the option **CDABLE_VARS** is set, and a parameter named *arg* exists whose value begins with a slash, treat its value as the directory. In that case, the parameter is added to the named directory hash table.

The second form of **cd** substitutes the string *new* for the string *old* in the name of the current directory, and tries to change to this new directory.

The third form of **cd** extracts an entry from the directory stack, and changes to that directory. An argument of the form ‘+*n*’ identifies a stack entry by counting from the left of the list shown by the **dirs** command, starting with zero. An argument of the form ‘-*n*’ counts from the right. If the **PUSHD_MINUS** option is set, the meanings of ‘+’ and ‘-’ in this context are swapped. If the **POSIX_CD** option is set, this form of **cd** is not recognised and will be interpreted as the first form.

If the **-q** (quiet) option is specified, the hook function **chpwd** and the functions in the array

chpwd_functions are not called. This is useful for calls to **cd** that do not change the environment seen by an interactive user.

If the **-s** option is specified, **cd** refuses to change the current directory if the given pathname contains symlinks. If the **-P** option is given or the **CHASE_LINKS** option is set, symbolic links are resolved to their true values. If the **-L** option is given symbolic links are retained in the directory (and not resolved) regardless of the state of the **CHASE_LINKS** option.

chdir Same as **cd**.

clone See the section ‘The zsh/clone Module’ in *zshmodules(1)*.

command [**-pvV**] *simple command*

The simple command argument is taken as an external command instead of a function or builtin and is executed. If the **POSIX_BUILTINS** option is set, builtins will also be executed but certain special properties of them are suppressed. The **-p** flag causes a default path to be searched instead of that in **\$path**. With the **-v** flag, **command** is similar to **whence** and with **-V**, it is equivalent to **whence -v**.

See also the section ‘Precommand Modifiers’ in *zshmisc(1)*.

comparguments

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

compcall

See the section ‘The zsh/comptcl Module’ in *zshmodules(1)*.

compctl

See the section ‘The zsh/comptcl Module’ in *zshmodules(1)*.

compdescribe

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

compfiles

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

compgroups

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

compquote

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

comptags

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

comptrry

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

compvalues

See the section ‘The zsh/computil Module’ in *zshmodules(1)*.

continue [*n*]

Resume the next iteration of the enclosing **for**, **while**, **until**, **select** or **repeat** loop. If an arithmetic expression *n* is specified, break out of *n*–1 loops and resume at the *n*th enclosing loop.

declare Same as **typeset**.

dirs [**-c**] [*arg* ...]

dirs [**-lpv**]

With no arguments, print the contents of the directory stack. Directories are added to this stack with the **pushd** command, and removed with the **cd** or **popd** commands. If arguments are specified, load them onto the directory stack, replacing anything that was there, and push the current directory onto the stack.

-c clear the directory stack.

- l** print directory names in full instead of using of using `~` expressions (see *Dynamic* and *Static named directories* in *zshexpn(1)*).
- p** print directory entries one per line.
- v** number the directories in the stack when printing.

disable [**-afmprs**] *name* ...

Temporarily disable the *named* hash table elements or patterns. The default is to disable builtin commands. This allows you to use an external command with the same name as a builtin command. The **-a** option causes **disable** to act on regular or global aliases. The **-s** option causes **disable** to act on suffix aliases. The **-f** option causes **disable** to act on shell functions. The **-r** options causes **disable** to act on reserved words. Without arguments all disabled hash table elements from the corresponding hash table are printed. With the **-m** flag the arguments are taken as patterns (which should be quoted to prevent them from undergoing filename expansion), and all hash table elements from the corresponding hash table matching these patterns are disabled. Disabled objects can be enabled with the **enable** command.

With the option **-p**, *name* ... refer to elements of the shell's pattern syntax as described in the section 'Filename Generation'. Certain elements can be disabled separately, as given below.

Note that patterns not allowed by the current settings for the options **EXTENDED_GLOB**, **KSH_GLOB** and **SH_GLOB** are never enabled, regardless of the setting here. For example, if **EXTENDED_GLOB** is not active, the pattern `^` is ineffective even if '**disable -p "^"**' has not been issued. The list below indicates any option settings that restrict the use of the pattern. It should be noted that setting **SH_GLOB** has a wider effect than merely disabling patterns as certain expressions, in particular those involving parentheses, are parsed differently.

The following patterns may be disabled; all the strings need quoting on the command line to prevent them from being interpreted immediately as patterns and the patterns are shown below in single quotes as a reminder.

- '?'** The pattern character `?` wherever it occurs, including when preceding a parenthesis with **KSH_GLOB**.
- '*'** The pattern character `*` wherever it occurs, including recursive globbing and when preceding a parenthesis with **KSH_GLOB**.
- '['** Character classes.
- '<' (NO_SH_GLOB)**
Numeric ranges.
- '|' (NO_SH_GLOB)**
Alternation in grouped patterns, case statements, or **KSH_GLOB** parenthesised expressions.
- '(' (NO_SH_GLOB)**
Grouping using single parentheses. Disabling this does not disable the use of parentheses for **KSH_GLOB** where they are introduced by a special character, nor for glob qualifiers (use '**setopt NO_BARE_GLOB_QUAL**' to disable glob qualifiers that use parentheses only).
- '~' (EXTENDED_GLOB)**
Exclusion in the form `A~B`.
- '^' (EXTENDED_GLOB)**
Exclusion in the form `A^B`.
- '#' (EXTENDED_GLOB)**
The pattern character `#` wherever it occurs, both for repetition of a previous pattern and for indicating globbing flags.

'?' (KSH_GLOB)

The grouping form ?(...). Note this is also disabled if '*' is disabled.

'*' (KSH_GLOB)

The grouping form *(...). Note this is also disabled if '*' is disabled.

'+' (KSH_GLOB)

The grouping form +(...).

'!' (KSH_GLOB)

The grouping form !(...).

'@' (KSH_GLOB)

The grouping form @(...).

disown [job ...]

job ... &|

job ... &!

Remove the specified *jobs* from the job table; the shell will no longer report their status, and will not complain if you try to exit an interactive shell with them running or stopped. If no *job* is specified, disown the current job.

If the *jobs* are currently stopped and the **AUTO_CONTINUE** option is not set, a warning is printed containing information about how to make them running after they have been disowned. If one of the latter two forms is used, the *jobs* will automatically be made running, independent of the setting of the **AUTO_CONTINUE** option.

echo [-neE] [arg ...]

Write each *arg* on the standard output, with a space separating each one. If the **-n** flag is not present, print a newline at the end. **echo** recognizes the following escape sequences:

\a	bell character
\b	backspace
\c	suppress subsequent characters and final newline
\e	escape
\f	form feed
\n	linefeed (newline)
\r	carriage return
\t	horizontal tab
\v	vertical tab
\\\	backslash
\0NNN	character code in octal
\xNN	character code in hexadecimal
\uNNNN	unicode character code in hexadecimal
\UNNNNNNNNN	unicode character code in hexadecimal

The **-E** flag, or the **BSD_ECHO** option, can be used to disable these escape sequences. In the latter case, **-e** flag can be used to enable them.

Note that for standards compliance a double dash does not terminate option processing; instead, it is printed directly. However, a single dash does terminate option processing, so the first dash, possibly following options, is not printed, but everything following it is printed as an argument. The single dash behaviour is different from other shells. For a more portable way of printing text, see **printf**, and for a more controllable way of printing text within zsh, see **print**.

echotc See the section ‘The zsh/termcap Module’ in *zshmodules(1)*.

echoti See the section ‘The zsh/terminfo Module’ in *zshmodules(1)*.

emulate [**-ILR**] [{**zsh|sh|ksh|csh**} [*flags* ...]]

Without any argument print current emulation mode.

With single argument set up zsh options to emulate the specified shell as much as possible. **csh** will never be fully emulated. If the argument is not one of the shells listed above, **zsh** will be used as a default; more precisely, the tests performed on the argument are the same as those used to determine the emulation at startup based on the shell name, see the section COMPATIBILITY in *zsh(1)*. In addition to setting shell options, the command also restores the pristine state of pattern enables, as if all patterns had been enabled using **enable -p**.

If the **emulate** command occurs inside a function that has been marked for execution tracing with **functions -t** then the **xtrace** option will be turned on regardless of emulation mode or other options. Note that code executed inside the function by the **..**, **source**, or **eval** commands is not considered to be running directly from the function, hence does not provoke this behaviour.

If the **-R** switch is given, all settable options are reset to their default value corresponding to the specified emulation mode, except for certain options describing the interactive environment; otherwise, only those options likely to cause portability problems in scripts and functions are altered. If the **-L** switch is given, the options **LOCAL_OPTIONS**, **LOCAL_PATTERNS** and **LOCAL_TRAPS** will be set as well, causing the effects of the **emulate** command and any **setopt**, **disable -p** or **enable -p**, and **trap** commands to be local to the immediately surrounding shell function, if any; normally these options are turned off in all emulation modes except **ksh**. The **-L** switch is mutually exclusive with the use of **-c** in *flags*.

If there is a single argument and the **-I** switch is given, the options that would be set or unset (the latter indicated with the prefix ‘**no**’) are listed. **-I** can be combined with **-L** or **-R** and the list will be modified in the appropriate way. Note the list does not depend on the current setting of options, i.e. it includes all options that may in principle change, not just those that would actually change.

The *flags* may be any of the invocation-time flags described in the section INVOCATION in *zsh(1)*, except that ‘**-o EMACS**’ and ‘**-o VI**’ may not be used. Flags such as ‘**+r**’/‘**+o RESTRICTED**’ may be prohibited in some circumstances.

If **-c arg** appears in *flags*, *arg* is evaluated while the requested emulation is temporarily in effect. In this case the emulation mode and all options are restored to their previous values before **emulate** returns. The **-R** switch may precede the name of the shell to emulate; note this has a meaning distinct from including **-R** in *flags*.

Use of **-c** enables ‘sticky’ emulation mode for functions defined within the evaluated expression: the emulation mode is associated thereafter with the function so that whenever the function is executed the emulation (respecting the **-R** switch, if present) and all options are set (and pattern disables cleared) before entry to the function, and the state is restored after exit. If the function is called when the sticky emulation is already in effect, either within an ‘**emulate shell -c**’ expression or within another function with the same sticky emulation, entry and exit from the function do not cause options to be altered (except due to standard processing such as the **LOCAL_OPTIONS** option). This also applies to functions marked for autoload within the sticky emulation; the appropriate set of options will be applied at the point the function is loaded as well as when it is run.

For example:

```
emulate sh -c 'fni() { setopt cshnullglob; }
fno() { fni; }
fno
```

The two functions **fni** and **fno** are defined with sticky **sh** emulation. **fno** is then executed, causing options associated with emulations to be set to their values in **sh**. **fno** then calls **fni**; because **fni** is also marked for sticky **sh** emulation, no option changes take place on entry to or exit from it. Hence the option **cshnullglob**, turned off by **sh** emulation, will be turned on within **fni** and remain on return to **fno**. On exit from **fno**, the emulation mode and all options will be restored to the state

they were in before entry to the temporary emulation.

The documentation above is typically sufficient for the intended purpose of executing code designed for other shells in a suitable environment. More detailed rules follow.

1. The sticky emulation environment provided by ‘**emulate shell -c**’ is identical to that provided by entry to a function marked for sticky emulation as a consequence of being defined in such an environment. Hence, for example, the sticky emulation is inherited by subfunctions defined within functions with sticky emulation.
2. No change of options takes place on entry to or exit from functions that are not marked for sticky emulation, other than those that would normally take place, even if those functions are called within sticky emulation.
3. No special handling is provided for functions marked for **autoload** nor for functions present in wordcode created by the **zcompile** command.
4. The presence or absence of the **-R** switch to **emulate** corresponds to different sticky emulation modes, so for example ‘**emulate sh -c**’, ‘**emulate -R sh -c**’ and ‘**emulate csh -c**’ are treated as three distinct sticky emulations.
5. Difference in shell options supplied in addition to the basic emulation also mean the sticky emulations are different, so for example ‘**emulate zsh -c**’ and ‘**emulate zsh -o cbases -c**’ are treated as distinct sticky emulations.

enable [**-afmprs**] *name* ...

Enable the *named* hash table elements, presumably disabled earlier with **disable**. The default is to enable builtin commands. The **-a** option causes **enable** to act on regular or global aliases. The **-s** option causes **enable** to act on suffix aliases. The **-f** option causes **enable** to act on shell functions. The **-r** option causes **enable** to act on reserved words. Without arguments all enabled hash table elements from the corresponding hash table are printed. With the **-m** flag the arguments are taken as patterns (should be quoted) and all hash table elements from the corresponding hash table matching these patterns are enabled. Enabled objects can be disabled with the **disable** builtin command.

enable -p reenables patterns disabled with **disable -p**. Note that it does not override globbing options; for example, ‘**enable -p "~~"**’ does not cause the pattern character **~** to be active unless the **EXTENDED_GLOB** option is also set. To enable all possible patterns (so that they may be individually disabled with **disable -p**), use ‘**setopt EXTENDED_GLOB KSH_GLOB NO_SH_GLOB**’.

eval [*arg* ...]

Read the arguments as input to the shell and execute the resulting command(s) in the current shell process. The return status is the same as if the commands had been executed directly by the shell; if there are no *args* or they contain no commands (i.e. are an empty string or whitespace) the return status is zero.

exec [**-cl**] [**-a** *argv0*] [*command* [*arg* ...]]

Replace the current shell with *command* rather than forking. If *command* is a shell builtin command or a shell function, the shell executes it, and exits when the command is complete.

With **-c** clear the environment; with **-l** prepend **-** to the **argv[0]** string of the command executed (to simulate a login shell); with **-a** *argv0* set the **argv[0]** string of the command executed. See the section ‘Precommand Modifiers’ in *zshmisc(1)*.

If the option **POSIX_BUILTINS** is set, *command* is never interpreted as a shell builtin command or shell function. This means further precommand modifiers such as **builtin** and **noglob** are also not interpreted within the shell. Hence *command* is always found by searching the command path.

If *command* is omitted but any redirections are specified, then the redirections will take effect in the current shell.

exit [*n*]

Exit the shell with the exit status specified by an arithmetic expression *n*; if none is specified, use the exit status from the last command executed. An EOF condition will also cause the shell to

exit, unless the **IGNORE_EOF** option is set.

See notes at the end of the section JOBS in *zshmisc(1)* for some possibly unexpected interactions of the **exit** command with jobs.

export [*name[=value]* ...]

The specified *names* are marked for automatic export to the environment of subsequently executed commands. Equivalent to **typeset -gx**. If a parameter specified does not already exist, it is created in the global scope.

false [*arg* ...]

Do nothing and return an exit status of 1.

fc [**-e** *ename*] [**-LI**] [**-m** *match*] [*old=new* ...] [*first* [*last*]]

fc **-l** [**-LI**] [**-nrdfEiD**] [**-t** *timefmt*] [**-m** *match*]

[*old=new* ...] [*first* [*last*]]

fc **-p** [**-a**] [*filename* [*histsize* [*savehistsize*]]]

fc **-P**

fc **-ARWI** [*filename*]

The **fc** command controls the interactive history mechanism. Note that reading and writing of history options is only performed if the shell is interactive. Usually this is detected automatically, but it can be forced by setting the **interactive** option when starting the shell.

The first two forms of this command select a range of events from *first* to *last* from the history list. The arguments *first* and *last* may be specified as a number or as a string. A negative number is used as an offset to the current history event number. A string specifies the most recent event beginning with the given string. All substitutions *old=new*, if any, are then performed on the text of the events.

In addition to the number range,

-I restricts to only internal events (not from **\$HISTFILE**)

-L restricts to only local events (not from other shells, see **SHARE_HISTORY** in *zshoptions(1)* — note that **\$HISTFILE** is considered local when read at startup)

-m takes the first argument as a pattern (should be quoted) and only the history events matching this pattern are considered

If *first* is not specified, it will be set to -1 (the most recent event), or to -16 if the **-I** flag is given. If *last* is not specified, it will be set to *first*, or to -1 if the **-I** flag is given. However, if the current event has added entries to the history with '**print -s**' or '**fc -R**', then the default *last* for **-I** includes all new history entries since the current event began.

When the **-I** flag is given, the resulting events are listed on standard output. Otherwise the editor program specified by **-e** *ename* is invoked on a file containing these history events. If **-e** is not given, the value of the parameter **FCEDIT** is used; if that is not set the value of the parameter **EDITOR** is used; if that is not set a builtin default, usually '**vi**' is used. If *ename* is ' $-$ ', no editor is invoked. When editing is complete, the edited command is executed.

The flag **-r** reverses the order of the events and the flag **-n** suppresses event numbers when listing.

Also when listing,

-d prints timestamps for each event

-f prints full time–date stamps in the US '*MM/DD/YY hh:mm*' format

-E prints full time–date stamps in the European '*dd.mm.yyyy hh:mm*' format

-i prints full time–date stamps in ISO8601 '*yyyy-mm-dd hh:mm*' format

-t *fmt* prints time and date stamps in the given format; *fmt* is formatted with the `strftime` function with the zsh extensions described for the **%D{string}** prompt format in the section EXPANSION OF PROMPT SEQUENCES in *zshmisc(1)*. The resulting formatted string must be no more than 256 characters or will not be printed

-D prints elapsed times; may be combined with one of the options above

‘**fc -p**’ pushes the current history list onto a stack and switches to a new history list. If the **-a** option is also specified, this history list will be automatically popped when the current function scope is exited, which is a much better solution than creating a trap function to call ‘**fc -P**’ manually. If no arguments are specified, the history list is left empty, **\$HISTFILE** is unset, and **\$HISTSIZE** & **\$SAVEHIST** are set to their default values. If one argument is given, **\$HISTFILE** is set to that filename, **\$HISTSIZE** & **\$SAVEHIST** are left unchanged, and the history file is read in (if it exists) to initialize the new list. If a second argument is specified, **\$HISTSIZE** & **\$SAVEHIST** are instead set to the single specified numeric value. Finally, if a third argument is specified, **\$SAVEHIST** is set to a separate value from **\$HISTSIZE**. You are free to change these environment values for the new history list however you desire in order to manipulate the new history list.

‘**fc -P**’ pops the history list back to an older list saved by ‘**fc -p**’. The current list is saved to its **\$HISTFILE** before it is destroyed (assuming that **\$HISTFILE** and **\$SAVEHIST** are set appropriately, of course). The values of **\$HISTFILE**, **\$HISTSIZE**, and **\$SAVEHIST** are restored to the values they had when ‘**fc -p**’ was called. Note that this restoration can conflict with making these variables “local”, so your best bet is to avoid local declarations for these variables in functions that use ‘**fc -p**’. The one other guaranteed-safe combination is declaring these variables to be local at the top of your function and using the automatic option (**-a**) with ‘**fc -p**’. Finally, note that it is legal to manually pop a push marked for automatic popping if you need to do so before the function exits.

‘**fc -R**’ reads the history from the given file, ‘**fc -W**’ writes the history out to the given file, and ‘**fc -A**’ appends the history out to the given file. If no filename is specified, the **\$HISTFILE** is assumed. If the **-I** option is added to **-R**, only those events that are not already contained within the internal history list are added. If the **-I** option is added to **-A** or **-W**, only those events that are new since last incremental append/write to the history file are appended/written. In any case, the created file will have no more than **\$SAVEHIST** entries.

fg [job ...]

job ... Bring each specified *job* in turn to the foreground. If no *job* is specified, resume the current job.

float [{+|-}Hghlprtux] [{+|-}EFLRZ [*n*]] [*name[=value]* ...]

Equivalent to **typeset -E**, except that options irrelevant to floating point numbers are not permitted.

functions [{+|-}UkmtTuWz] [-x *num*] [*name* ...]

functions -c *oldfn newfn*

functions -M [-s] *mathfn* [*min* [*max* [*shellfn*]]]

functions -M [-m *pattern* ...]

functions +M [-m] *mathfn* ...

Equivalent to **typeset -f**, with the exception of the **-c**, **-x**, **-M** and **-W** options. For **functions -u** and **functions -U**, see **autoload**, which provides additional options.

The **-x** option indicates that any functions output will have each leading tab for indentation, added by the shell to show syntactic structure, expanded to the given number *num* of spaces. *num* can also be 0 to suppress all indentation.

The **-W** option turns on the option **WARN_NESTED_VAR** for the named function or functions only. The option is turned off at the start of nested functions (apart from anonymous functions) unless the called function also has the **-W** attribute.

The **-c** option causes *oldfn* to be copied to *newfn*. The copy is efficiently handled internally by reference counting. If *oldfn* was marked for autoload it is first loaded and if this fails the copy fails. Either function may subsequently be redefined without affecting the other. A typical idiom is that *oldfn* is the name of a library shell function which is then redefined to call **newfn**, thereby installing a modified version of the function.

Use of the **-M** option may not be combined with any of the options handled by **typeset -f**.

functions -M *mathfn* defines *mathfn* as the name of a mathematical function recognised in all forms of arithmetical expressions; see the section ‘Arithmetic Evaluation’ in *zshmisc(1)*. By default *mathfn* may take any number of comma-separated arguments. If *min* is given, it must have exactly *min* args; if *min* and *max* are both given, it must have at least *min* and at most *max* args. *max* may be -1 to indicate that there is no upper limit.

By default the function is implemented by a shell function of the same name; if *shellfn* is specified it gives the name of the corresponding shell function while *mathfn* remains the name used in arithmetical expressions. The name of the function in **\$0** is *mathfn* (not *shellfn* as would usually be the case), provided the option **FUNCTION_ARGZERO** is in effect. The positional parameters in the shell function correspond to the arguments of the mathematical function call. The result of the last arithmetical expression evaluated inside the shell function (even if it is a form that normally only returns a status) gives the result of the mathematical function.

If the additional option **-s** is given to **functions -M**, the argument to the function is a single string: anything between the opening and matching closing parenthesis is passed to the function as a single argument, even if it includes commas or white space. The minimum and maximum argument specifiers must therefore be 1 if given. An empty argument list is passed as a zero-length string.

functions -M with no arguments lists all such user-defined functions in the same form as a definition. With the additional option **-m** and a list of arguments, all functions whose *mathfn* matches one of the pattern arguments are listed.

function +M removes the list of mathematical functions; with the additional option **-m** the arguments are treated as patterns and all functions whose *mathfn* matches the pattern are removed. Note that the shell function implementing the behaviour is not removed (regardless of whether its name coincides with *mathfn*).

For example, the following prints the cube of 3 :

```
zmath_cube() { (( $1 * $1 * $1 )) }
functions -M cube 1 1 zmath_cube
print $(( cube(3) ))
```

The following string function takes a single argument, including the commas, so prints 11 :

```
stringfn() { (( $#1 )) }
functions -Ms stringfn
print $(( stringfn(foo,bar,rod) ))
```

getcap See the section ‘The zsh/cap Module’ in *zshmodules(1)*.

getln [**-AclneE**] *name* ...

Read the top value from the buffer stack and put it in the shell parameter *name*. Equivalent to **read -zr**.

getopts *optstring name [arg ...]*

Checks the *args* for legal options. If the *args* are omitted, use the positional parameters. A valid option argument begins with a $+$ or a $-$. An argument not beginning with a $+$ or a $-$, or the argument $--$, ends the options. Note that a single $-$ is not considered a valid option argument. *optstring* contains the letters that **getopts** recognizes. If a letter is followed by a $:$, that option requires an argument. The options can be separated from the argument by blanks.

Each time it is invoked, **getopts** places the option letter it finds in the shell parameter *name*, prepended with a $+$ when *arg* begins with a $+$. The index of the next *arg* is stored in **OPTIND**. The option argument, if any, is stored in **OPTARG**.

The first option to be examined may be changed by explicitly assigning to **OPTIND**. **OPTIND** has an initial value of **1**, and is normally set to **1** upon entry to a shell function and restored upon exit (this is disabled by the **POSIX_BUILTINS** option). **OPTARG** is not reset and retains its value from the most recent call to **getopts**. If either of **OPTIND** or **OPTARG** is explicitly unset,

it remains unset, and the index or option argument is not stored. The option itself is still stored in *name* in this case.

A leading ‘:’ in *optstring* causes **getopts** to store the letter of any invalid option in **OPTARG**, and to set *name* to ‘?’ for an unknown option and to ‘:’ when a required argument is missing. Otherwise, **getopts** sets *name* to ‘?’ and prints an error message when an option is invalid. The exit status is nonzero when there are no more options.

hash [**-LdfmrV**] [*name[=value]*] ...

hash can be used to directly modify the contents of the command hash table, and the named directory hash table. Normally one would modify these tables by modifying one’s **PATH** (for the command hash table) or by creating appropriate shell parameters (for the named directory hash table). The choice of hash table to work on is determined by the **-d** option; without the option the command hash table is used, and with the option the named directory hash table is used.

A command *name* starting with a / is never hashed, whether by explicit use of the **hash** command or otherwise. Such a command is always found by direct look up in the file system.

Given no arguments, and neither the **-r** or **-f** options, the selected hash table will be listed in full.

The **-r** option causes the selected hash table to be emptied. It will be subsequently rebuilt in the normal fashion. The **-f** option causes the selected hash table to be fully rebuilt immediately. For the command hash table this hashes all the absolute directories in the **PATH**, and for the named directory hash table this adds all users’ home directories. These two options cannot be used with any arguments.

The **-m** option causes the arguments to be taken as patterns (which should be quoted) and the elements of the hash table matching those patterns are printed. This is the only way to display a limited selection of hash table elements.

For each *name* with a corresponding *value*, put ‘*name*’ in the selected hash table, associating it with the pathname ‘*value*’. In the command hash table, this means that whenever ‘*name*’ is used as a command argument, the shell will try to execute the file given by ‘*value*’. In the named directory hash table, this means that ‘*value*’ may be referred to as ‘~*name*’.

For each *name* with no corresponding *value*, attempt to add *name* to the hash table, checking what the appropriate **value** is in the normal manner for that hash table. If an appropriate **value** can’t be found, then the hash table will be unchanged.

The **-v** option causes hash table entries to be listed as they are added by explicit specification. If has no effect if used with **-f**.

If the **-L** flag is present, then each hash table entry is printed in the form of a call to **hash**.

history Same as **fc -l**.

integer [{+|-}Hghlpertux] [{+|-}LRZi [*n*]] [*name[=value]*] ...]

Equivalent to **typeset -i**, except that options irrelevant to integers are not permitted.

jobs [**-dlprs**] [*job* ...]

jobs **-Z** *string*

Lists information about each given job, or all jobs if *job* is omitted. The **-l** flag lists process IDs, and the **-p** flag lists process groups. If the **-r** flag is specified only running jobs will be listed and if the **-s** flag is given only stopped jobs are shown. If the **-d** flag is given, the directory from which the job was started (which may not be the current directory of the job) will also be shown.

The **-Z** option replaces the shell’s argument and environment space with the given string, truncated if necessary to fit. This will normally be visible in **ps** (*ps(1)*) listings. This feature is typically used by daemons, to indicate their state.

kill [**-s** *signal_name* | **-n** *signal_number* | **-sig**] *job* ...

kill -l [*sig* ...]

Sends either **SIGTERM** or the specified signal to the given jobs or processes. Signals are given by number or by names, with or without the ‘**SIG**’ prefix. If the signal being sent is not ‘**KILL**’ or ‘**CONT**’, then the job will be sent a ‘**CONT**’ signal if it is stopped. The argument *job* can be the process ID of a job not in the job list. In the second form, **kill -l**, if *sig* is not specified the signal names are listed. Otherwise, for each *sig* that is a name, the corresponding signal number is listed. For each *sig* that is a signal number or a number representing the exit status of a process which was terminated or stopped by a signal the name of the signal is printed.

On some systems, alternative signal names are allowed for a few signals. Typical examples are **SIGCHLD** and **SIGCLD** or **SIGPOLL** and **SIGIO**, assuming they correspond to the same signal number. **kill -l** will only list the preferred form, however **kill -l alt** will show if the alternative form corresponds to a signal number. For example, under Linux **kill -l IO** and **kill -l POLL** both output 29, hence **kill -IO** and **kill -POLL** have the same effect.

Many systems will allow process IDs to be negative to kill a process group or zero to kill the current process group.

let *arg* ...

Evaluate each *arg* as an arithmetic expression. See the section ‘Arithmetic Evaluation’ in *zshmisc(1)* for a description of arithmetic expressions. The exit status is 0 if the value of the last expression is nonzero, 1 if it is zero, and 2 if an error occurred.

limit [-hs] [*resource* [*limit*]] ...

Set or display resource limits. Unless the **-s** flag is given, the limit applies only the children of the shell. If **-s** is given without other arguments, the resource limits of the current shell is set to the previously set resource limits of the children.

If *limit* is not specified, print the current limit placed on *resource*, otherwise set the limit to the specified value. If the **-h** flag is given, use hard limits instead of soft limits. If no *resource* is given, print all limits.

When looping over multiple resources, the shell will abort immediately if it detects a badly formed argument. However, if it fails to set a limit for some other reason it will continue trying to set the remaining limits.

resource can be one of:

addressspace

Maximum amount of address space used.

aiomemorylocked

Maximum amount of memory locked in RAM for AIO operations.

aiooperations

Maximum number of AIO operations.

cachedthreads

Maximum number of cached threads.

coredumpsize

Maximum size of a core dump.

cputime

Maximum CPU seconds per process.

datasize

Maximum data size (including stack) for each process.

descriptors

Maximum value for a file descriptor.

filesize

Largest single file allowed.

kqueues

Maximum number of kqueues allocated.

maxproc

Maximum number of processes.

maxthreads

Maximum number of threads per process.

memorylocked

Maximum amount of memory locked in RAM.

memoryuse

Maximum resident set size.

msgqueue

Maximum number of bytes in POSIX message queues.

posixlocks

Maximum number of POSIX locks per user.

pseudoterminals

Maximum number of pseudo-terminals.

resident

Maximum resident set size.

sigpending

Maximum number of pending signals.

sockbufsize

Maximum size of all socket buffers.

stacksize

Maximum stack size for each process.

swapsize

Maximum amount of swap used.

vmemorysize

Maximum amount of virtual memory.

Which of these resource limits are available depends on the system. *resource* can be abbreviated to any unambiguous prefix. It can also be an integer, which corresponds to the integer defined for the resource by the operating system.

If argument corresponds to a number which is out of the range of the resources configured into the shell, the shell will try to read or write the limit anyway, and will report an error if this fails. As the shell does not store such resources internally, an attempt to set the limit will fail unless the **-s** option is present.

limit is a number, with an optional scaling factor, as follows:

n h	hours
n k	kilobytes (default)
n m	megabytes or minutes
n g	gigabytes
[m m :] s	minutes and seconds

The **limit** command is not made available by default when the shell starts in a mode emulating another shell. It can be made available with the command ‘**zmodload -F zsh/rlimits b:limit**’.

local [{+|-}A{HUhlp}rtux] [{+|-}EFLRZi [*n*]] [*name*[=value] ...]

Same as **typeset**, except that the options **-g**, and **-f** are not permitted. In this case the **-x** option does not force the use of **-g**, i.e. exported variables will be local to functions.

log List all users currently logged in who are affected by the current setting of the **watch** parameter.

logout [*n*]

Same as **exit**, except that it only works in a login shell.

noglob *simple command*

See the section ‘Precommand Modifiers’ in *zshmisc*(1).

popd [**-q**] [{+|-}n]

Remove an entry from the directory stack, and perform a **cd** to the new top directory. With no argument, the current top entry is removed. An argument of the form '+n' identifies a stack entry by counting from the left of the list shown by the **dirs** command, starting with zero. An argument of the form '-n' counts from the right. If the **PUSHD_MINUS** option is set, the meanings of '+' and '-' in this context are swapped.

If the **-q** (quiet) option is specified, the hook function **chpwd** and the functions in the array **\$chpwd_functions** are not called, and the new directory stack is not printed. This is useful for calls to **popd** that do not change the environment seen by an interactive user.

print [**-abcDilmnNoOpPrsSz**] [**-u n**] [**-f format**] [**-C cols**]
[**-v name**] [**-xX tabstop**] [**-R** [**-en**]] [**arg ...**]

With the '**-f**' option the arguments are printed as described by **printf**. With no flags or with the flag '**-**', the arguments are printed on the standard output as described by **echo**, with the following differences: the escape sequence '**\M-x**' (or '**\Mx**') metafies the character *x* (sets the highest bit), '**\C-x**' (or '**\Cx**') produces a control character ('**\C-@**' and '**\C-?**' give the characters NULL and delete), a character code in octal is represented by '**\NNN**' (instead of '**\0NNN**'), and '**\E**' is a synonym for '**\e**'. Finally, if not in an escape sequence, '****' escapes the following character and is not printed.

-a Print arguments with the column incrementing first. Only useful with the **-c** and **-C** options.

-b Recognize all the escape sequences defined for the **bindkey** command, see the section 'Zle Builtins' in *zshzle(1)*.

-c Print the arguments in columns. Unless **-a** is also given, arguments are printed with the row incrementing first.

-C cols

Print the arguments in *cols* columns. Unless **-a** is also given, arguments are printed with the row incrementing first.

-D Treat the arguments as paths, replacing directory prefixes with **~** expressions corresponding to directory names, as appropriate.

-i If given together with **-o** or **-O**, sorting is performed case-independently.

-l Print the arguments separated by newlines instead of spaces. Note: if the list of arguments is empty, **print -l** will still output one empty line. To print a possibly-empty list of arguments one per line, use **print -C1**, as in '**print -rC1 -- "\$list[@]"**'.

-m Take the first argument as a pattern (should be quoted), and remove it from the argument list together with subsequent arguments that do not match this pattern.

-n Do not add a newline to the output.

-N Print the arguments separated and terminated by nulls. Again, **print -rNC1 -- "\$list[@]"** is a canonical way to print an arbitrary list as null-delimited records.

-o Print the arguments sorted in ascending order.

-O Print the arguments sorted in descending order.

-p Print the arguments to the input of the coprocess.

-P Perform prompt expansion (see EXPANSION OF PROMPT SEQUENCES in *zshmisc(1)*). In combination with '**-f**', prompt escape sequences are parsed only within interpolated arguments, not within the format string.

-r Ignore the escape conventions of **echo**.

-R Emulate the BSD **echo** command, which does not process escape sequences unless the **-e** flag is given. The **-n** flag suppresses the trailing newline. Only the **-e** and **-n** flags are

recognized after **-R**; all other arguments and options are printed.

- s** Place the results in the history list instead of on the standard output. Each argument to the **print** command is treated as a single word in the history, regardless of its content.
- S** Place the results in the history list instead of on the standard output. In this case only a single argument is allowed; it will be split into words as if it were a full shell command line. The effect is similar to reading the line from a history file with the **HIST_LEX_WORDS** option active.
- u n** Print the arguments to file descriptor *n*.
- v name** Store the printed arguments as the value of the parameter *name*.
- x tab-stop** Expand leading tabs on each line of output in the printed string assuming a tab stop every *tab-stop* characters. This is appropriate for formatting code that may be indented with tabs. Note that leading tabs of any argument to print, not just the first, are expanded, even if **print** is using spaces to separate arguments (the column count is maintained across arguments but may be incorrect on output owing to previous unexpanded tabs).
The start of the output of each **print** command is assumed to be aligned with a tab stop. Widths of multibyte characters are handled if the option **MULTIBYTE** is in effect. This option is ignored if other formatting options are in effect, namely column alignment or **printf** style, or if output is to a special location such as shell history or the command line editor.
- X tab-stop** This is similar to **-x**, except that all tabs in the printed string are expanded. This is appropriate if tabs in the arguments are being used to produce a table format.
- z** Push the arguments onto the editing buffer stack, separated by spaces.
If any of '**-m**', '**-o**' or '**-O**' are used in combination with '**-f**' and there are no arguments (after the removal process in the case of '**-m**') then nothing is printed.

printf [**-v** *name*] *format* [*arg ...*]

Print the arguments according to the format specification. Formatting rules are the same as used in C. The same escape sequences as for **echo** are recognised in the format. All C conversion specifications ending in one of **cstdiouxXeEfgGn** are handled. In addition to this, '**%b**' can be used instead of '**%s**' to cause escape sequences in the argument to be recognised and '**%q**' can be used to quote the argument in such a way that allows it to be reused as shell input. With the numeric format specifiers, if the corresponding argument starts with a quote character, the numeric value of the following character is used as the number to print; otherwise the argument is evaluated as an arithmetic expression. See the section 'Arithmetic Evaluation' in *zshmisc(1)* for a description of arithmetic expressions. With '**%n**', the corresponding argument is taken as an identifier which is created as an integer parameter.

Normally, conversion specifications are applied to each argument in order but they can explicitly specify the *n*th argument is to be used by replacing '**%**' by '**%on\$**' and '*****' by '***n\$**'. It is recommended that you do not mix references of this explicit style with the normal style and the handling of such mixed styles may be subject to future change.

If arguments remain unused after formatting, the format string is reused until all arguments have been consumed. With the **print** builtin, this can be suppressed by using the **-r** option. If more arguments are required by the format than have been specified, the behaviour is as if zero or an empty string had been specified as the argument.

The **-v** option causes the output to be stored as the value of the parameter *name*, instead of printed. If *name* is an array and the format string is reused when consuming arguments then one array element will be used for each use of the format string.

pushd [**-qsLP**] [*arg*]
pushd [**-qsLP**] *old new*
pushd [**-qsLP**] {+|-}n

Change the current directory, and push the old current directory onto the directory stack. In the first form, change the current directory to *arg*. If *arg* is not specified, change to the second directory on the stack (that is, exchange the top two entries), or change to **\$HOME** if the **PUSHD_TO_HOME** option is set or if there is only one entry on the stack. Otherwise, *arg* is interpreted as it would be by **cd**. The meaning of *old* and *new* in the second form is also the same as for **cd**.

The third form of **pushd** changes directory by rotating the directory list. An argument of the form ‘+n’ identifies a stack entry by counting from the left of the list shown by the **dirs** command, starting with zero. An argument of the form ‘-n’ counts from the right. If the **PUSHD_MINUS** option is set, the meanings of ‘+’ and ‘-’ in this context are swapped.

If the **-q** (quiet) option is specified, the hook function **chpwd** and the functions in the array **\$chpwd_functions** are not called, and the new directory stack is not printed. This is useful for calls to **pushd** that do not change the environment seen by an interactive user.

If the option **-q** is not specified and the shell option **PUSHD_SILENT** is not set, the directory stack will be printed after a **pushd** is performed.

The options **-s**, **-L** and **-P** have the same meanings as for the **cd** builtin.

pushln [*arg ...*]

Equivalent to **print -nz**.

pwd [**-rLP**]

Print the absolute pathname of the current working directory. If the **-r** or the **-P** flag is specified, or the **CHASE_LINKS** option is set and the **-L** flag is not given, the printed path will not contain symbolic links.

r Same as **fc -e -**.

read [**-rszpqAclnE**] [**-t** [*num*]] [**-k** [*num*]] [**-d** *delim*]
[**-u** *n*] [*name*[?*prompt*]] [*name ...*]

Read one line and break it into fields using the characters in **\$IFS** as separators, except as noted below. The first field is assigned to the *firstname*, the second field to the second *name*, etc., with leftover fields assigned to the last *name*. If *name* is omitted then **REPLY** is used for scalars and **reply** for arrays.

-r Raw mode: a ‘\’ at the end of a line does not signify line continuation and backslashes in the line don’t quote the following character and are not removed.

-s Don’t echo back characters if reading from the terminal.

-q Read only one character from the terminal and set *name* to ‘y’ if this character was ‘y’ or ‘Y’ and to ‘n’ otherwise. With this flag set the return status is zero only if the character was ‘y’ or ‘Y’. This option may be used with a timeout (see **-t**); if the read times out, or encounters end of file, status 2 is returned. Input is read from the terminal unless one of **-u** or **-p** is present. This option may also be used within zle widgets.

-k [*num*]

Read only one (or *num*) characters. All are assigned to the first *name*, without word splitting. This flag is ignored when **-q** is present. Input is read from the terminal unless one of **-u** or **-p** is present. This option may also be used within zle widgets.

Note that despite the mnemonic ‘key’ this option does read full characters, which may consist of multiple bytes if the option **MULTIBYTE** is set.

-z Read one entry from the editor buffer stack and assign it to the first *name*, without word splitting. Text is pushed onto the stack with ‘**print -z**’ or with **push-line** from the line

editor (see *zshzle(1)*). This flag is ignored when the **-k** or **-q** flags are present.

- e**
- E** The input read is printed (echoed) to the standard output. If the **-e** flag is used, no input is assigned to the parameters.
- A** The first *name* is taken as the name of an array and all words are assigned to it.
- c**
- l** These flags are allowed only if called inside a function used for completion (specified with the **-K** flag to **compctl**). If the **-c** flag is given, the words of the current command are read. If the **-l** flag is given, the whole line is assigned as a scalar. If both flags are present, **-l** is used and **-c** is ignored.
- n** Together with **-c**, the number of the word the cursor is on is read. With **-l**, the index of the character the cursor is on is read. Note that the command name is word number 1, not word 0, and that when the cursor is at the end of the line, its character index is the length of the line plus one.
- u** *n* Input is read from file descriptor *n*.
- p** Input is read from the coprocess.
- d** *delim* Input is terminated by the first character of *delim* instead of by newline.
- t** [*num*] Test if input is available before attempting to read. If *num* is present, it must begin with a digit and will be evaluated to give a number of seconds, which may be a floating point number; in this case the read times out if input is not available within this time. If *num* is not present, it is taken to be zero, so that **read** returns immediately if no input is available. If no input is available, return status 1 and do not set any variables.

This option is not available when reading from the editor buffer with **-z**, when called from within completion with **-c** or **-l**, with **-q** which clears the input queue before reading, or within zle where other mechanisms should be used to test for input.

Note that **read** does not attempt to alter the input processing mode. The default mode is canonical input, in which an entire line is read at a time, so usually ‘**read -t**’ will not read anything until an entire line has been typed. However, when reading from the terminal with **-k** input is processed one key at a time; in this case, only availability of the first character is tested, so that e.g. ‘**read -t -k 2**’ can still block on the second character. Use two instances of ‘**read -t -k**’ if this is not what is wanted.

If the first argument contains a ‘?’ , the remainder of this word is used as a *prompt* on standard error when the shell is interactive.

The value (exit status) of **read** is 1 when an end-of-file is encountered, or when **-c** or **-l** is present and the command is not called from a **compctl** function, or as described for **-q**. Otherwise the value is 0.

The behavior of some combinations of the **-k**, **-p**, **-q**, **-u** and **-z** flags is undefined. Presently **-q** cancels all the others, **-p** cancels **-u**, **-k** cancels **-z**, and otherwise **-z** cancels both **-p** and **-u**.

The **-c** or **-l** flags cancel any and all of **-kpquz**.

readonly

Same as **typeset -r**. With the **POSIX_BUILTINS** option set, same as **typeset -gr**.

rehash

Same as **hash -r**.

return [*n*]

Causes a shell function or ‘.’ script to return to the invoking script with the return status specified by an arithmetic expression *n*. If *n* is omitted, the return status is that of the last command executed.

If **return** was executed from a trap in a **TRAPNAL** function, the effect is different for zero and non-zero return status. With zero status (or after an implicit return at the end of the trap), the shell will return to whatever it was previously processing; with a non-zero status, the shell will behave as interrupted except that the return status of the trap is retained. Note that the numeric value of the signal which caused the trap is passed as the first argument, so the statement ‘**return \$((128+\$1))**’ will return the same status as if the signal had not been trapped.

sched See the section ‘The zsh/sched Module’ in *zshmodules(1)*.

set [{+|-}options | {+|-}o [*option_name*]] ... [{+|-}A [*name*]]
[*arg* ...]

Set the options for the shell and/or set the positional parameters, or declare and set an array. If the **-s** option is given, it causes the specified arguments to be sorted before assigning them to the positional parameters (or to the array *name* if **-A** is used). With **+s** sort arguments in descending order. For the meaning of the other flags, see *zshoptions(1)*. Flags may be specified by name using the **-o** option. If no option name is supplied with **-o**, the current option states are printed: see the description of **setopt** below for more information on the format. With **+o** they are printed in a form that can be used as input to the shell.

If the **-A** flag is specified, *name* is set to an array containing the given *args*; if no *name* is specified, all arrays are printed together with their values.

If **+A** is used and *name* is an array, the given arguments will replace the initial elements of that array; if no *name* is specified, all arrays are printed without their values.

The behaviour of arguments after **-A** *name* or **+A** *name* depends on whether the option **KSH_ARRAYS** is set. If it is not set, all arguments following *name* are treated as values for the array, regardless of their form. If the option is set, normal option processing continues at that point; only regular arguments are treated as values for the array. This means that

set -A array -x -- foo

sets **array** to ‘**-x -- foo**’ if **KSH_ARRAYS** is not set, but sets the array to **foo** and turns on the option ‘**-x**’ if it is set.

If the **-A** flag is not present, but there are arguments beyond the options, the positional parameters are set. If the option list (if any) is terminated by ‘**--**’, and there are no further arguments, the positional parameters will be unset.

If no arguments and no ‘**--**’ are given, then the names and values of all parameters are printed on the standard output. If the only argument is ‘**+**’, the names of all parameters are printed.

For historical reasons, ‘**set -**’ is treated as ‘**set +xv**’ and ‘**set - args**’ as ‘**set +xv -- args**’ when in any other emulation mode than zsh’s native mode.

setcap See the section ‘The zsh/cap Module’ in *zshmodules(1)*.

setopt [{+|-}options | {+|-}o *option_name*] [**-m**] [*name* ...]

Set the options for the shell. All options specified either with flags or by name are set.

If no arguments are supplied, the names of all options currently set are printed. The form is chosen so as to minimize the differences from the default options for the current emulation (the default emulation being native **zsh**, shown as **<Z>** in *zshoptions(1)*). Options that are on by default for the emulation are shown with the prefix **no** only if they are off, while other options are shown without the prefix **no** and only if they are on. In addition to options changed from the default state by the user, any options activated automatically by the shell (for example, **SHIN_STDIN** or **INTERACTIVE**) will be shown in the list. The format is further modified by the option

KSH_OPTION_PRINT, however the rationale for choosing options with or without the **no** prefix remains the same in this case.

If the **-m** flag is given the arguments are taken as patterns (which should be quoted to protect them from filename expansion), and all options with names matching these patterns are set.

Note that a bad option name does not cause execution of subsequent shell code to be aborted; this behaviour is different from that of '**set -o**'. This is because **set** is regarded as a special builtin by the POSIX standard, but **setopt** is not.

shift [**-p**] [*n*] [*name ...*]

The positional parameters $\${n+1} \dots$ are renamed to **\$1** ..., where *n* is an arithmetic expression that defaults to 1. If any *names* are given then the arrays with these names are shifted instead of the positional parameters.

If the option **-p** is given arguments are instead removed (popped) from the end rather than the start of the array.

source *file* [*arg ...*]

Same as '.', except that the current directory is always searched and is always searched first, before directories in **\$path**.

stat See the section 'The zsh/stat Module' in *zshmodules*(1).

suspend [**-f**]

Suspend the execution of the shell (send it a **SIGTSTP**) until it receives a **SIGCONT**. Unless the **-f** option is given, this will refuse to suspend a login shell.

test [*arg ...*]

[[*arg ...*]]

Like the system version of **test**. Added for compatibility; use conditional expressions instead (see the section 'Conditional Expressions'). The main differences between the conditional expression syntax and the **test** and [builtins are: these commands are not handled syntactically, so for example an empty variable expansion may cause an argument to be omitted; syntax errors cause status 2 to be returned instead of a shell error; and arithmetic operators expect integer arguments rather than arithmetic expressions.

The command attempts to implement POSIX and its extensions where these are specified. Unfortunately there are intrinsic ambiguities in the syntax; in particular there is no distinction between test operators and strings that resemble them. The standard attempts to resolve these for small numbers of arguments (up to four); for five or more arguments compatibility cannot be relied on. Users are urged wherever possible to use the '[[...]]' test syntax which does not have these ambiguities.

times Print the accumulated user and system times for the shell and for processes run from the shell.

trap [*arg*] [*sig ...*]

arg is a series of commands (usually quoted to protect it from immediate evaluation by the shell) to be read and executed when the shell receives any of the signals specified by one or more *sig* args. Each *sig* can be given as a number, or as the name of a signal either with or without the string **SIG** in front (e.g. 1, HUP, and SIGHUP are all the same signal).

If *arg* is '−', then the specified signals are reset to their defaults, or, if no *sig* args are present, all traps are reset.

If *arg* is an empty string, then the specified signals are ignored by the shell (and by the commands it invokes).

If *arg* is omitted but one or more *sig* args are provided (i.e. the first argument is a valid signal number or name), the effect is the same as if *arg* had been specified as '−'.

The **trap** command with no arguments prints a list of commands associated with each signal.

If *sig* is **ZERR** then *arg* will be executed after each command with a nonzero exit status. **ERR** is

an alias for **ZERR** on systems that have no **SIGERR** signal (this is the usual case).

If *sig* is **DEBUG** then *arg* will be executed before each command if the option **DEBUG_BEFORE_CMD** is set (as it is by default), else after each command. Here, a ‘command’ is what is described as a ‘sublist’ in the shell grammar, see the section SIMPLE COMMANDS & PIPELINES in *zshmisc(1)*. If **DEBUG_BEFORE_CMD** is set various additional features are available. First, it is possible to skip the next command by setting the option **ERR_EXIT**; see the description of the **ERR_EXIT** option in *zshoptions(1)*. Also, the shell parameter **ZSH_DEBUG_CMD** is set to the string corresponding to the command to be executed following the trap. Note that this string is reconstructed from the internal format and may not be formatted the same way as the original text. The parameter is unset after the trap is executed.

If *sig* is **0** or **EXIT** and the **trap** statement is executed inside the body of a function, then the command *arg* is executed after the function completes. The value of **\$?** at the start of execution is the exit status of the shell or the return status of the function exiting. If *sig* is **0** or **EXIT** and the **trap** statement is not executed inside the body of a function, then the command *arg* is executed when the shell terminates; the trap runs before any **zshexit** hook functions.

ZERR, **DEBUG**, and **EXIT** traps are not executed inside other traps. **ZERR** and **DEBUG** traps are kept within subshells, while other traps are reset.

Note that traps defined with the **trap** builtin are slightly different from those defined as ‘**TRAP-NAL () { ... }**’, as the latter have their own function environment (line numbers, local variables, etc.) while the former use the environment of the command in which they were called. For example,

```
trap 'print $LINENO' DEBUG
```

will print the line number of a command executed after it has run, while

```
TRAPDEBUG() { print $LINENO; }
```

will always print the number zero.

Alternative signal names are allowed as described under **kill** above. Defining a trap under either name causes any trap under an alternative name to be removed. However, it is recommended that for consistency users stick exclusively to one name or another.

true [*arg* ...]

Do nothing and return an exit status of 0.

ttyctl [**-fu**]

The **-f** option freezes the tty (i.e. terminal or terminal emulator), and **-u** unfreezes it. When the tty is frozen, no changes made to the tty settings by external programs will be honored by the shell, except for changes in the size of the screen; the shell will simply reset the settings to their previous values as soon as each command exits or is suspended. Thus, **stty** and similar programs have no effect when the tty is frozen. Freezing the tty does not cause the current state to be remembered: instead, it causes future changes to the state to be blocked.

Without options it reports whether the terminal is frozen or not.

Note that, regardless of whether the tty is frozen or not, the shell needs to change the settings when the line editor starts, so unfreezing the tty does not guarantee settings made on the command line are preserved. Strings of commands run between editing the command line will see a consistent tty state. See also the shell variable **STTY** for a means of initialising the tty before running external commands.

type [**-wfpamsS**] *name* ...

Equivalent to **whence -v**.

```
typeset [ {+|-}AHUaghlmrtux ] [ {+|-}EFLRZip [ n ] ]
```

```
[ + ] [ name[=value] ... ]
```

```
typeset -T [ {+|-}Uglrx ] [ {+|-}LRZp [ n ] ]
```

```
[ + | SCALAR[=value] array[=(value ...)] [ sep ] ]
```

```
typeset -f [ {+|-}TUKmtuz ] [ + ] [ name ... ]
```

Set or display attributes and values for shell parameters.

Except as noted below for control flags that change the behavior, a parameter is created for each *name* that does not already refer to one. When inside a function, a new parameter is created for every *name* (even those that already exist), and is unset again when the function completes. See ‘Local Parameters’ in *zshparam(1)*. The same rules apply to special shell parameters, which retain their special attributes when made local.

For each *name*=*value* assignment, the parameter *name* is set to *value*.

If the shell option **TYPESET_SILENT** is not set, for each remaining *name* that refers to a parameter that is already set, the name and value of the parameter are printed in the form of an assignment. Nothing is printed for newly-created parameters, or when any attribute flags listed below are given along with the *name*. Using ‘+’ instead of minus to introduce an attribute turns it off.

If no *name* is present, the names and values of all parameters are printed. In this case the attribute flags restrict the display to only those parameters that have the specified attributes, and using ‘+’ rather than ‘-’ to introduce the flag suppresses printing of the values of parameters when there is no parameter name.

All forms of the command handle scalar assignment. Array assignment is possible if any of the reserved words **declare**, **export**, **float**, **integer**, **local**, **readonly** or **typeset** is matched when the line is parsed (N.B. not when it is executed). In this case the arguments are parsed as assignments, except that the ‘+=’ syntax and the **GLOB_ASSIGN** option are not supported, and scalar values after = are *not* split further into words, even if expanded (regardless of the setting of the **KSH_TYPESET** option; this option is obsolete).

Examples of the differences between command and reserved word parsing:

```
# Reserved word parsing
typeset svar=$(echo one word) avar=(several words)
```

The above creates a scalar parameter **svar** and an array parameter **avar** as if the assignments had been

```
svar="one word"
avar=(several words)
```

On the other hand:

```
# Normal builtin interface
builtin typeset svar=$(echo two words)
```

The **builtin** keyword causes the above to use the standard builtin interface to **typeset** in which argument parsing is performed in the same way as for other commands. This example creates a scalar **svar** containing the value **two** and another scalar parameter **words** with no value. An array value in this case would either cause an error or be treated as an obscure set of glob qualifiers.

Arbitrary arguments are allowed if they take the form of assignments after command line expansion; however, these only perform scalar assignment:

```
var='svar=val'
typeset $var
```

The above sets the scalar parameter **svar** to the value **val**. Parentheses around the value within **var** would not cause array assignment as they will be treated as ordinary characters when **\$var** is substituted. Any non-trivial expansion in the name part of the assignment causes the argument to be treated in this fashion:

typeset {var1,var2,var3}=name

The above syntax is valid, and has the expected effect of setting the three parameters to the same value, but the command line is parsed as a set of three normal command line arguments to **typeset** after expansion. Hence it is not possible to assign to multiple arrays by this means.

Note that each interface to any of the commands may be disabled separately. For example, ‘**disable -r typeset**’ disables the reserved word interface to **typeset**, exposing the builtin interface, while ‘**disable typeset**’ disables the builtin. Note that disabling the reserved word interface for **typeset** may cause problems with the output of ‘**typeset -p**’, which assumes the reserved word interface is available in order to restore array and associative array values.

Unlike parameter assignment statements, **typeset**’s exit status on an assignment that involves a command substitution does not reflect the exit status of the command substitution. Therefore, to test for an error in a command substitution, separate the declaration of the parameter from its initialization:

```
# WRONG
typeset var1=$(exit 1) || echo "Trouble with var1"
```

```
# RIGHT
typeset var1 && var1=$(exit 1) || echo "Trouble with var1"
```

To initialize a parameter *param* to a command output and mark it readonly, use **typeset -r param** or **readonly param** after the parameter assignment statement.

If no attribute flags are given, and either no *name* arguments are present or the flag **+m** is used, then each parameter name printed is preceded by a list of the attributes of that parameter (**array**, **association**, **exported**, **float**, **integer**, **readonly**, or **undefined** for autoloaded parameters not yet loaded). If **+m** is used with attrib ute flags, and all those flags are introduced with **+**, the matching parameter names are printed but their values are not.

The following control flags change the behavior of **typeset**:

- +** If ‘**+**’ appears by itself in a separate word as the last option, then the names of all parameters (functions with **-f**) are printed, but the values (function bodies) are not. No *name* arguments may appear, and it is an error for any other options to follow ‘**+**’. The effect of ‘**+**’ is as if all attribute flags which precede it were given with a ‘**+**’ prefix. For example, ‘**typeset -U +**’ is equivalent to ‘**typeset +U**’ and displays the names of all arrays having the uniqueness attribute, whereas ‘**typeset -f -U +**’ displays the names of all autoloadable functions. If **+** is the only option, then type information (array, readonly, etc.) is also printed for each parameter, in the same manner as ‘**typeset +m ".*"**’.

- g** The **-g** (global) means that any resulting parameter will not be restricted to local scope. Note that this does not necessarily mean that the parameter will be global, as the flag will apply to any existing parameter (even if unset) from an enclosing function. This flag does not affect the parameter after creation, hence it has no effect when listing existing parameters, nor does the flag **+g** have any effect except in combination with **-m** (see below).

- m** If the **-m** flag is given the *name* arguments are taken as patterns (use quoting to prevent these from being interpreted as file patterns). With no attribute flags, all parameters (or functions with the **-f** flag) with matching names are printed (the shell option **TYPESET_SILENT** is not used in this case).

If the **+g** flag is combined with **-m**, a new local parameter is created for every matching parameter that is not already local. Otherwise **-m** applies all other flags or assignments to the existing parameters.

Except when assignments are made with *name=value*, using **+m** forces the matching parameters and their attributes to be printed, even inside a function. Note that **-m** is ignored if no patterns are given, so ‘**typeset -m**’ displays attributes but ‘**typeset -a +m**’ does not.

-p [n] If the **-p** option is given, parameters and values are printed in the form of a typeset command with an assignment, regardless of other flags and options. Note that the **-H** flag on parameters is respected; no value will be shown for these parameters.

-p may be followed by an optional integer argument. Currently only the value **1** is supported. In this case arrays and associative arrays are printed with newlines between indented elements for readability.

-T [scalar[=value] array[=(value ...)] [sep]]

This flag has a different meaning when used with **-f**; see below. Otherwise the **-T** option requires zero, two, or three arguments to be present. With no arguments, the list of parameters created in this fashion is shown. With two or three arguments, the first two are the name of a scalar and of an array parameter (in that order) that will be tied together in the manner of **\$PATH** and **\$path**. The optional third argument is a single-character separator which will be used to join the elements of the array to form the scalar; if absent, a colon is used, as with **\$PATH**. Only the first character of the separator is significant; any remaining characters are ignored. Multibyte characters are not yet supported.

Only one of the scalar and array parameters may be assigned an initial value (the restrictions on assignment forms described above also apply).

Both the scalar and the array may be manipulated as normal. If one is unset, the other will automatically be unset too. There is no way of untying the variables without unsetting them, nor of converting the type of one of them with another **typeset** command; **+T** does not work, assigning an array to *scalar* is an error, and assigning a scalar to *array* sets it to be a single-element array.

Note that both ‘**typeset -xT ...**’ and ‘**export -T ...**’ work, but only the scalar will be marked for export. Setting the value using the scalar version causes a split on all separators (which cannot be quoted). It is possible to apply **-T** to two previously tied variables but with a different separator character, in which case the variables remain joined as before but the separator is changed.

When an existing scalar is tied to a new array, the value of the scalar is preserved but no attribute other than **export** will be preserved.

Attribute flags that transform the final value (**-L**, **-R**, **-Z**, **-I**, **-u**) are only applied to the expanded value at the point of a parameter expansion expression using ‘\$’. They are not applied when a parameter is retrieved internally by the shell for any purpose.

The following attribute flags may be specified:

-A The names refer to associative array parameters; see ‘Array Parameters’ in *zshparam(1)*.

-L [n]

Left justify and remove leading blanks from the value when the parameter is expanded. If *n* is nonzero, it defines the width of the field. If *n* is zero, the width is determined by the width of the value of the first assignment. In the case of numeric parameters, the length of the complete value assigned to the parameter is used to determine the width, not the value that would be output.

The width is the count of characters, which may be multibyte characters if the **MULTIBYTE** option is in effect. Note that the screen width of the character is not taken into account; if this is required, use padding with parameter expansion flags **\$(ml...)** as described in ‘Parameter Expansion Flags’ in *zshexpn(1)*.

When the parameter is expanded, it is filled on the right with blanks or truncated if necessary to fit the field. Note truncation can lead to unexpected results with numeric parameters. Leading zeros are removed if the **-Z** flag is also set.

-R [n]

Similar to **-L**, except that right justification is used; when the parameter is expanded, the field is left filled with blanks or truncated from the end. May not be combined with the **-Z** flag.

-U

For arrays (but not for associative arrays), keep only the first occurrence of each duplicated value. This may also be set for tied parameters (see **-T**) or colon-separated special parameters like **PATH** or **IGNORE**, etc. Note the flag takes effect on assignment, and the type of the variable being assigned to is determinative; for variables with shared values it is therefore recommended to set the flag for all interfaces, e.g. ‘**typeset -U PATH path**’.

This flag has a different meaning when used with **-f**; see below.

-Z [n]

Specially handled if set along with the **-L** flag. Otherwise, similar to **-R**, except that leading zeros are used for padding instead of blanks if the first non-blank character is a digit. Numeric parameters are specially handled: they are always eligible for padding with zeroes, and the zeroes are inserted at an appropriate place in the output.

-a

The names refer to array parameters. An array parameter may be created this way, but it may be assigned to in the **typeset** statement only if the reserved word form of **typeset** is enabled (as it is by default). When displaying, both normal and associative arrays are shown.

-f

The names refer to functions rather than parameters. No assignments can be made, and the only other valid flags are **-t**, **-T**, **-k**, **-u**, **-U** and **-z**. The flag **-t** turns on execution tracing for this function; the flag **-T** does the same, but turns off tracing for any named (not anonymous) function called from the present one, unless that function also has the **-t** or **-T** flag. The **-u** and **-U** flags cause the function to be marked for autoloading; **-U** also causes alias expansion to be suppressed when the function is loaded. See the description of the ‘**autoload**’ builtin for details.

Note that the builtin **functions** provides the same basic capabilities as **typeset -f** but gives access to a few extra options; **autoload** gives further additional options for the case **typeset -fu** and **typeset -fU**.

-h

Hide: only useful for special parameters (those marked ‘<S>’ in the table in *zsh-param(1)*), and for local parameters with the same name as a special parameter, though harmless for others. A special parameter with this attribute will not retain its special effect when made local. Thus after ‘**typeset -h PATH**’, a function containing ‘**typeset PATH**’ will create an ordinary local parameter without the usual behaviour of **PATH**. Alternatively, the local parameter may itself be given this attribute; hence inside a function ‘**typeset -h PATH**’ creates an ordinary local parameter and the special **PATH** parameter is not altered in any way. It is also possible to create a local parameter using ‘**typeset +h special**’, where the local copy of *special* will retain its special properties regardless of having the **-h** attribute. Global special parameters loaded from shell modules (currently those in **zsh/mapfile** and **zsh/parameter**) are automatically given the **-h** attribute to avoid name clashes.

-H

Hide value: specifies that **typeset** will not display the value of the parameter when listing parameters; the display for such parameters is always as if the ‘+’ flag had been given. Use of the parameter is in other respects normal, and the option does not apply if the parameter is specified by name, or by pattern with the **-m** option. This is on by default for the parameters in the **zsh/parameter** and **zsh/mapfile** modules. Note, however, that unlike the **-h** flag this is also useful for non-special parameters.

-i [n]

Use an internal integer representation. If *n* is nonzero it defines the output arithmetic base, otherwise it is determined by the first assignment. Bases from 2 to 36 inclusive are allowed.

-E [n]

Use an internal double-precision floating point representation. On output the variable will be converted to scientific notation. If *n* is nonzero it defines the number of significant figures to display; the default is ten.

-F [n] Use an internal double-precision floating point representation. On output the variable will be converted to fixed-point decimal notation. If *n* is nonzero it defines the number of digits to display after the decimal point; the default is ten.

-l Convert the result to lower case whenever the parameter is expanded. The value is *not* converted when assigned.

-r The given *names* are marked readonly. Note that if *name* is a special parameter, the readonly attribute can be turned on, but cannot then be turned off.

If the **POSIX_BUILTINS** option is set, the readonly attribute is more restrictive: unset variables can be marked readonly and cannot then be set; furthermore, the readonly attribute cannot be removed from any variable.

It is still possible to change other attributes of the variable though, some of which like **-U** or **-Z** would affect the value. More generally, the readonly attribute should not be relied on as a security mechanism.

Note that in zsh (like in pdksh but unlike most other shells) it is still possible to create a local variable of the same name as this is considered a different variable (though this variable, too, can be marked readonly). Special variables that have been made readonly retain their value and readonly attribute when made local.

-t Tags the named parameters. Tags have no special meaning to the shell. This flag has a different meaning when used with **-f**; see above.

-u Convert the result to upper case whenever the parameter is expanded. The value is *not* converted when assigned. This flag has a different meaning when used with **-f**; see above.

-x Mark for automatic export to the environment of subsequently executed commands. If the option **GLOBAL_EXPORT** is set, this implies the option **-g**, unless **+g** is also explicitly given; in other words the parameter is not made local to the enclosing function. This is for compatibility with previous versions of zsh.

ulimit [-HSa] [{ -bcdflmnpqrsTtvwx | -N resource } [limit] ...]

Set or display resource limits of the shell and the processes started by the shell. The value of *limit* can be a number in the unit specified below or one of the values '**unlimited**', which removes the limit on the resource, or '**hard**', which uses the current value of the hard limit on the resource.

By default, only soft limits are manipulated. If the **-H** flag is given use hard limits instead of soft limits. If the **-S** flag is given together with the **-H** flag set both hard and soft limits.

If no options are used, the file size limit (**-f**) is assumed.

If *limit* is omitted the current value of the specified resources are printed. When more than one resource value is printed, the limit name and unit is printed before each value.

When looping over multiple resources, the shell will abort immediately if it detects a badly formed argument. However, if it fails to set a limit for some other reason it will continue trying to set the remaining limits.

Not all the following resources are supported on all systems. Running **ulimit -a** will show which are supported.

-a Lists all of the current resource limits.

-b Socket buffer size in bytes (N.B. not kilobytes)

-c 512-byte blocks on the size of core dumps.

-d	Kilobytes on the size of the data segment.
-f	512-byte blocks on the size of files written.
-i	The number of pending signals.
-k	The number of kqueues allocated.
-l	Kilobytes on the size of locked-in memory.
-m	Kilobytes on the size of physical memory.
-n	open file descriptors.
-p	The number of pseudo-terminals.
-q	Bytes in POSIX message queues.
-r	Maximum real time priority. On some systems where this is not available, such as NetBSD, this has the same effect as -T for compatibility with sh .
-s	Kilobytes on the size of the stack.
-T	The number of simultaneous threads available to the user.
-t	CPU seconds to be used.
-u	The number of processes available to the user.
-v	Kilobytes on the size of virtual memory. On some systems this refers to the limit called ‘address space’.
-w	Kilobytes on the size of swapped out memory.
-x	The number of locks on files.

A resource may also be specified by integer in the form ‘**-N resource**’, where *resource* corresponds to the integer defined for the resource by the operating system. This may be used to set the limits for resources known to the shell which do not correspond to option letters. Such limits will be shown by number in the output of ‘**ulimit -a**’.

The number may alternatively be out of the range of limits compiled into the shell. The shell will try to read or write the limit anyway, and will report an error if this fails.

umask [-S] [mask]

The umask is set to *mask*. *mask* can be either an octal number or a symbolic value as described in *chmod(1)*. If *mask* is omitted, the current value is printed. The **-S** option causes the mask to be printed as a symbolic value. Otherwise, the mask is printed as an octal number. Note that in the symbolic form the permissions you specify are those which are to be allowed (not denied) to the users specified.

unalias [-ams] name ...

Removes aliases. This command works the same as **unhash -a**, except that the **-a** option removes all regular or global aliases, or with **-s** all suffix aliases: in this case no *name* arguments may appear. The options **-m** (remove by pattern) and **-s** without **-a** (remove listed suffix aliases) behave as for **unhash -a**. Note that the meaning of **-a** is different between **unalias** and **unhash**.

unfunction

Same as **unhash -f**.

unhash [-adfms] name ...

Remove the element named *name* from an internal hash table. The default is remove elements from the command hash table. The **-a** option causes **unhash** to remove regular or global aliases; note when removing a global aliases that the argument must be quoted to prevent it from being expanded before being passed to the command. The **-s** option causes **unhash** to remove suffix aliases. The **-f** option causes **unhash** to remove shell functions. The **-d** option causes **unhash** to remove named directories. If the **-m** flag is given the arguments are taken as patterns (should be quoted) and all elements of the corresponding hash table with matching names will be removed.

unlimit [-hs] resource ...

The resource limit for each *resource* is set to the hard limit. If the **-h** flag is given and the shell has appropriate privileges, the hard resource limit for each *resource* is removed. The resources of the shell process are only changed if the **-s** flag is given.

The **unlimit** command is not made available by default when the shell starts in a mode emulating

another shell. It can be made available with the command ‘**zmodload -F zsh/rlimits b:unlimit**’.

unset [**-fmv**] *name* ...

Each named parameter is unset. Local parameters remain local even if unset; they appear unset within scope, but the previous value will still reappear when the scope ends.

Individual elements of associative array parameters may be unset by using subscript syntax on *name*, which should be quoted (or the entire command prefixed with **noglob**) to protect the subscript from filename generation.

If the **-m** flag is specified the arguments are taken as patterns (should be quoted) and all parameters with matching names are unset. Note that this cannot be used when unsetting associative array elements, as the subscript will be treated as part of the pattern.

The **-v** flag specifies that *name* refers to parameters. This is the default behaviour.

unset -f is equivalent to **unfunction**.

unsetopt [{+|-}options | {+|-}o *option_name*] [*name* ...]

Unset the options for the shell. All options specified either with flags or by name are unset. If no arguments are supplied, the names of all options currently unset are printed. If the **-m** flag is given the arguments are taken as patterns (which should be quoted to preserve them from being interpreted as glob patterns), and all options with names matching these patterns are unset.

vared See the section ‘Zle Builtins’ in *zshzle(1)*.

wait [*job* ...]

Wait for the specified jobs or processes. If *job* is not given then all currently active child processes are waited for. Each *job* can be either a job specification or the process ID of a job in the job table. The exit status from this command is that of the job waited for. If *job* represents an unknown job or process ID, a warning is printed (unless the **POSIX_BUILTINS** option is set) and the exit status is 127.

It is possible to wait for recent processes (specified by process ID, not by job) that were running in the background even if the process has exited. Typically the process ID will be recorded by capturing the value of the variable **\$!** immediately after the process has been started. There is a limit on the number of process IDs remembered by the shell; this is given by the value of the system configuration parameter **CHILD_MAX**. When this limit is reached, older process IDs are discarded, least recently started processes first.

Note there is no protection against the process ID wrapping, i.e. if the wait is not executed soon enough there is a chance the process waited for is the wrong one. A conflict implies both process IDs have been generated by the shell, as other processes are not recorded, and that the user is potentially interested in both, so this problem is intrinsic to process IDs.

whence [**-vcwfamsS**] [**-x num**] *name* ...

For each *name*, indicate how it would be interpreted if used as a command name.

If *name* is not an alias, built-in command, external command, shell function, hashed command, or a reserved word, the exit status shall be non-zero, and -- if **-v**, **-c**, or **-w** was passed -- a message will be written to standard output. (This is different from other shells that write that message to standard error.)

whence is most useful when *name* is only the last path component of a command, i.e. does not include a ‘/’; in particular, pattern matching only succeeds if just the non-directory component of the command is passed.

-v Produce a more verbose report.

-c Print the results in a **csh**-like format. This takes precedence over **-v**.

-w For each *name*, print ‘*name*: *word*’ where *word* is one of **alias**, **builtin**, **command**, **function**, **hashed**, **reserved** or **none**, according as *name* corresponds to an alias, a built-in command, an external command, a shell function, a command defined with the **hash**

builtin, a reserved word, or is not recognised. This takes precedence over **-v** and **-c**.

- f** Causes the contents of a shell function to be displayed, which would otherwise not happen unless the **-c** flag were used.
- p** Do a path search for *name* even if it is an alias, reserved word, shell function or builtin.
- a** Do a search for all occurrences of *name* throughout the command path. Normally only the first occurrence is printed.
- m** The arguments are taken as patterns (pattern characters should be quoted), and the information is displayed for each command matching one of these patterns.
- s** If a pathname contains symlinks, print the symlink-free pathname as well.
- S** As **-s**, but if the pathname had to be resolved by following multiple symlinks, the intermediate steps are printed, too. The symlink resolved at each step might be anywhere in the path.
- x num** Expand tabs when outputting shell functions using the **-c** option. This has the same effect as the **-x** option to the **functions** builtin.

where [**-wpmsS**] [**-x num**] *name* ...

Equivalent to **whence -ca**.

which [**-wpamsS**] [**-x num**] *name* ...

Equivalent to **whence -c**.

zcompile [**-U**] [**-z** | **-k**] [**-R** | **-M**]*file* [*name* ...]

zcompile -ca [**-m**] [**-R** | **-M**]*file* [*name* ...]

zcompile -t*file* [*name* ...]

This builtin command can be used to compile functions or scripts, storing the compiled form in a file, and to examine files containing the compiled form. This allows faster autoloading of functions and sourcing of scripts by avoiding parsing of the text when the files are read.

The first form (without the **-c**, **-a** or **-t** options) creates a compiled file. If only the *file* argument is given, the output file has the name '*file.zwc*' and will be placed in the same directory as the *file*. The shell will load the compiled file instead of the normal function file when the function is autoloaded; see the section ‘Autoloading Functions’ in *zshmisc(1)* for a description of how autoloaded functions are searched. The extension **.zwc** stands for ‘zsh word code’.

If there is at least one *name* argument, all the named files are compiled into the output *file* given as the first argument. If *file* does not end in **.zwc**, this extension is automatically appended. Files containing multiple compiled functions are called ‘digest’ files, and are intended to be used as elements of the **FPATH/fpath** special array.

The second form, with the **-c** or **-a** options, writes the compiled definitions for all the named functions into *file*. For **-c**, the names must be functions currently defined in the shell, not those marked for autoloading. Undefined functions that are marked for autoloading may be written by using the **-a** option, in which case the **fpath** is searched and the contents of the definition files for those functions, if found, are compiled into *file*. If both **-c** and **-a** are given, names of both defined functions and functions marked for autoloading may be given. In either case, the functions in files written with the **-c** or **-a** option will be autoloaded as if the **KSH_AUTOLOAD** option were unset.

The reason for handling loaded and not-yet-loaded functions with different options is that some definition files for autoloading define multiple functions, including the function with the same name as the file, and, at the end, call that function. In such cases the output of ‘**zcompile -c**’ does not include the additional functions defined in the file, and any other initialization code in the file is lost. Using ‘**zcompile -a**’ captures all this extra information.

If the **-m** option is combined with **-c** or **-a**, the *names* are used as patterns and all functions whose names match one of these patterns will be written. If no *name* is given, the definitions of all

functions currently defined or marked as autoloaded will be written.

Note the second form cannot be used for compiling functions that include redirections as part of the definition rather than within the body of the function; for example

```
fn1() { { ... } >~/logfile }
```

can be compiled but

```
fn1() { ... } >~/logfile
```

cannot. It is possible to use the first form of **zcompile** to compile autoloadable functions that include the full function definition instead of just the body of the function.

The third form, with the **-t** option, examines an existing compiled file. Without further arguments, the names of the original files compiled into it are listed. The first line of output shows the version of the shell which compiled the file and how the file will be used (i.e. by reading it directly or by mapping it into memory). With arguments, nothing is output and the return status is set to zero if definitions for *all names* were found in the compiled file, and non-zero if the definition for at least one *name* was not found.

Other options:

- U** Aliases are not expanded when compiling the *named* files.
- R** When the compiled file is read, its contents are copied into the shell's memory, rather than memory-mapped (see **-M**). This happens automatically on systems that do not support memory mapping.
- When compiling scripts instead of autoloadable functions, it is often desirable to use this option; otherwise the whole file, including the code to define functions which have already been defined, will remain mapped, consequently wasting memory.
- M** The compiled file is mapped into the shell's memory when read. This is done in such a way that multiple instances of the shell running on the same host will share this mapped file. If neither **-R** nor **-M** is given, the **zcompile** builtin decides what to do based on the size of the compiled file.
- k**
- z** These options are used when the compiled file contains functions which are to be autoloaded. If **-z** is given, the function will be autoloaded as if the **KSH_AUTOLOAD** option is *not* set, even if it is set at the time the compiled file is read, while if the **-k** is given, the function will be loaded as if **KSH_AUTOLOAD** is set. These options also take precedence over any **-k** or **-z** options specified to the **autoload** builtin. If neither of these options is given, the function will be loaded as determined by the setting of the **KSH_AUTOLOAD** option at the time the compiled file is read.

These options may also appear as many times as necessary between the listed *names* to specify the loading style of all following functions, up to the next **-k** or **-z**.

The created file always contains two versions of the compiled format, one for big-endian machines and one for small-endian machines. The upshot of this is that the compiled file is machine independent and if it is read or mapped, only one half of the file is actually used (and mapped).

zformat

See the section 'The zsh/zutil Module' in *zshmodules*(1).

zftp See the section 'The zsh/zftp Module' in *zshmodules*(1).

zle See the section 'Zle Builtins' in *zshzle*(1).

```
zmodload [ -dL ] [ -s ] [ ... ]
zmodload -F [ -alLme -P param ] module [ [+−]feature ... ]
zmodload -e [ -A ] [ ... ]
zmodload [ -a [ -bcpf [ -I ] ] [ -iL ] ... ]
zmodload -u [ -abcdpf [ -I ] ] [ -iL ] ...
zmodload -A [ -L ] [ modalias[=module] ... ]
zmodload -R modalias ...
```

Performs operations relating to zsh's loadable modules. Loading of modules while the shell is running ('dynamical loading') is not available on all operating systems, or on all installations on a particular operating system, although the **zmodload** command itself is always available and can be used to manipulate modules built into versions of the shell executable without dynamical loading.

Without arguments the names of all currently loaded binary modules are printed. The **-L** option causes this list to be in the form of a series of **zmodload** commands. Forms with arguments are:

```
zmodload [ -is ] name ...
zmodload -u [ -i ] name ...
```

In the simplest case, **zmodload** loads a binary module. The module must be in a file with a name consisting of the specified *name* followed by a standard suffix, usually '.so' ('.sl' on HPUX). If the module to be loaded is already loaded the duplicate module is ignored. If **zmodload** detects an inconsistency, such as an invalid module name or circular dependency list, the current code block is aborted. If it is available, the module is loaded if necessary, while if it is not available, non-zero status is silently returned. The option **-i** is accepted for compatibility but has no effect.

The *named* module is searched for in the same way a command is, using **\$module_path** instead of **\$path**. However, the path search is performed even when the module name contains a '/', which it usually does. There is no way to prevent the path search.

If the module supports features (see below), **zmodload** tries to enable all features when loading a module. If the module was successfully loaded but not all features could be enabled, **zmodload** returns status 2.

If the option **-s** is given, no error is printed if the module was not available (though other errors indicating a problem with the module are printed). The return status indicates if the module was loaded. This is appropriate if the caller considers the module optional.

With **-u**, **zmodload** unloads modules. The same *name* must be given that was given when the module was loaded, but it is not necessary for the module to exist in the file system. The **-i** option suppresses the error if the module is already unloaded (or w as never loaded).

Each module has a boot and a cleanup function. The module will not be loaded if its boot function fails. Similarly a module can only be unloaded if its cleanup function runs successfully.

```
zmodload -F [ -almLe -P param ] module [ [+−]feature ... ]
```

zmodload **-F** allows more selective control over the features provided by modules. With no options apart from **-F**, the module named *module* is loaded, if it was not already loaded, and the list of *features* is set to the required state. If no *features* are specified, the module is loaded, if it was not already loaded, but the state of features is unchanged. Each feature may be preceded by a + to turn the feature on, or - to turn it off; the + is assumed if neither character is present. Any feature not explicitly mentioned is left in its current state; if the module was not previously loaded this means any such features will remain disabled. The return status is zero if all features were set, 1 if the module failed to load, and 2 if some features could not be set (for example, a parameter couldn't be added because there was a different parameter of the same name) but the module was loaded.

The standard features are builtins, conditions, parameters and math functions; these are indicated by the prefix '**b:**', '**c:**' ('**C:**' for an infix condition), '**p:**' and '**f:**', respectively,

followed by the name that the corresponding feature would have in the shell. For example, ‘**b:strftime**’ indicates a builtin named **strftime** and **p:EPOCHSECONDS** indicates a parameter named **EPOCHSECONDS**. The module may provide other (‘abstract’) features of its own as indicated by its documentation; these have no prefix.

With **-I** or **-L**, features provided by the module are listed. With **-I** alone, a list of features together with their states is shown, one feature per line. With **-L** alone, a **zmodload -F** command that would cause enabled features of the module to be turned on is shown. With **-IL**, a **zmodload -F** command that would cause all the features to be set to their current state is shown. If one of these combinations is given with the option **-P param** then the parameter *param* is set to an array of features, either features together with their state or (if **-L** alone is given) enabled features.

With the option **-L** the module name may be omitted; then a list of all enabled features for all modules providing features is printed in the form of **zmodload -F** commands. If **-I** is also given, the state of both enabled and disabled features is output in that form.

A set of features may be provided together with **-I** or **-L** and a module name; in that case only the state of those features is considered. Each feature may be preceded by + or – but the character has no effect. If no set of features is provided, all features are considered.

With **-e**, the command first tests that the module is loaded; if it is not, status 1 is returned. If the module is loaded, the list of features given as an argument is examined. Any feature given with no prefix is simply tested to see if the module provides it; any feature given with a prefix + or – is tested to see if it is provided and in the given state. If the tests on all features in the list succeed, status 0 is returned, else status 1.

With **-m**, each entry in the given list of features is taken as a pattern to be matched against the list of features provided by the module. An initial + or – must be given explicitly. This may not be combined with the **-a** option as autoloads must be specified explicitly.

With **-a**, the given list of features is marked for autoload from the specified module, which may not yet be loaded. An optional + may appear before the feature name. If the feature is prefixed with –, any existing autoload is removed. The options **-I** and **-L** may be used to list autoloads. Autoloading is specific to individual features; when the module is loaded only the requested feature is enabled. Autoload requests are preserved if the module is subsequently unloaded until an explicit ‘**zmodload -Fa module -feature**’ is issued. It is not an error to request an autoload for a feature of a module that is already loaded.

When the module is loaded each autoload is checked against the features actually provided by the module; if the feature is not provided the autoload request is deleted. A warning message is output; if the module is being loaded to provide a different feature, and that autoload is successful, there is no effect on the status of the current command. If the module is already loaded at the time when **zmodload -Fa** is run, an error message is printed and status 1 returned.

zmodload -Fa can be used with the **-I**, **-L**, **-e** and **-P** options for listing and testing the existence of autoloadable features. In this case **-I** is ignored if **-L** is specified. **zmodload -FaL** with no module name lists autoloads for all modules.

Note that only standard features as described above can be autoloaded; other features require the module to be loaded before enabling.

```
zmodload -d [ -L ] [ name ]
zmodload -d name dep ...
zmodload -ud name [ dep ... ]
```

The **-d** option can be used to specify module dependencies. The modules named in the second and subsequent arguments will be loaded before the module named in the first

argument.

With **-d** and one argument, all dependencies for that module are listed. With **-d** and no arguments, all module dependencies are listed. This listing is by default in a Makefile-like format. The **-L** option changes this format to a list of **zmodload -d** commands.

If **-d** and **-u** are both used, dependencies are removed. If only one argument is given, all dependencies for that module are removed.

```
zmodload -ab [ -L ]
zmodload -ab [ -i ] name [ builtin ... ]
zmodload -ub [ -i ] builtin ...
```

The **-ab** option defines autoloaded builtins. It defines the specified *builtins*. When any of those builtins is called, the module specified in the first argument is loaded and all its features are enabled (for selective control of features use '**zmodload -F -a**' as described above). If only the *name* is given, one builtin is defined, with the same name as the module. **-i** suppresses the error if the builtin is already defined or autoloaded, but not if another builtin of the same name is already defined.

With **-ab** and no arguments, all autoloaded builtins are listed, with the module name (if different) shown in parentheses after the builtin name. The **-L** option changes this format to a list of **zmodload -a** commands.

If **-b** is used together with the **-u** option, it removes builtins previously defined with **-ab**. This is only possible if the builtin is not yet loaded. **-i** suppresses the error if the builtin is already removed (or never existed).

Autoload requests are retained if the module is subsequently unloaded until an explicit '**zmodload -ub builtin**' is issued.

```
zmodload -ac [ -IL ]
zmodload -ac [ -iI ] name [ cond ... ]
zmodload -uc [ -iI ] cond ...
```

The **-ac** option is used to define autoloaded condition codes. The *cond* strings give the names of the conditions defined by the module. The optional **-I** option is used to define infix condition names. Without this option prefix condition names are defined.

If given no condition names, all defined names are listed (as a series of **zmodload** commands if the **-L** option is given).

The **-uc** option removes definitions for autoloaded conditions.

```
zmodload -ap [ -L ]
zmodload -ap [ -i ] name [ parameter ... ]
zmodload -up [ -i ] parameter ...
```

The **-p** option is like the **-b** and **-c** options, but makes **zmodload** work on autoloaded parameters instead.

```
zmodload -af [ -L ]
zmodload -af [ -i ] name [ function ... ]
zmodload -uf [ -i ] function ...
```

The **-f** option is like the **-b**, **-p**, and **-c** options, but makes **zmodload** work on autoloaded math functions instead.

```
zmodload -a [ -L ]
zmodload -a [ -i ] name [ builtin ... ]
zmodload -ua [ -i ] builtin ...
```

Equivalent to **-ab** and **-ub**.

```
zmodload -e [ -A ] [ string ... ]
```

The **-e** option without arguments lists all loaded modules; if the **-A** option is also given, module aliases corresponding to loaded modules are also shown. If arguments are

provided, nothing is printed; the return status is set to zero if all *strings* given as arguments are names of loaded modules and to one if at least one *string* is not the name of a loaded module. This can be used to test for the availability of things implemented by modules. In this case, any aliases are automatically resolved and the **-A** flag is not used.

zmodload -A [-L] [*modalias[=module]* ...]

For each argument, if both *modalias* and *module* are given, define *modalias* to be an alias for the module *module*. If the module *modalias* is ever subsequently requested, either via a call to **zmodload** or implicitly, the shell will attempt to load *module* instead. If *module* is not given, show the definition of *modalias*. If no arguments are given, list all defined module aliases. When listing, if the **-L** flag was also given, list the definition as a **zmodload** command to recreate the alias.

The existence of aliases for modules is completely independent of whether the name resolved is actually loaded as a module: while the alias exists, loading and unloading the module under any alias has exactly the same effect as using the resolved name, and does not affect the connection between the alias and the resolved name which can be removed either by **zmodload -R** or by redefining the alias. Chains of aliases (i.e. where the first resolved name is itself an alias) are valid so long as these are not circular. As the aliases take the same format as module names, they may include path separators: in this case, there is no requirement for any part of the path named to exist as the alias will be resolved first. For example, ‘**any/old/alias**’ is always a valid alias.

Dependencies added to aliased modules are actually added to the resolved module; these remain if the alias is removed. It is valid to create an alias whose name is one of the standard shell modules and which resolves to a different module. However, if a module has dependencies, it will not be possible to use the module name as an alias as the module will already be marked as a loadable module in its own right.

Apart from the above, aliases can be used in the **zmodload** command anywhere module names are required. However, aliases will not be shown in lists of loaded modules with a bare ‘**zmodload**’.

zmodload -R *modalias* ...

For each *modalias* argument that was previously defined as a module alias via **zmodload -A**, delete the alias. If any was not defined, an error is caused and the remainder of the line is ignored.

Note that **zsh** makes no distinction between modules that were linked into the shell and modules that are loaded dynamically. In both cases this builtin command has to be used to make available the builtins and other things defined by modules (unless the module is autoloaded on these definitions). This is true even for systems that don’t support dynamic loading of modules.

zparseopts

See the section ‘The zsh/zutil Module’ in *zshmodules(1)*.

zprof See the section ‘The zsh/zprof Module’ in *zshmodules(1)*.

zpty See the section ‘The zsh/zpty Module’ in *zshmodules(1)*.

zregexecparse

See the section ‘The zsh/zutil Module’ in *zshmodules(1)*.

zsocket See the section ‘The zsh/net/socket Module’ in *zshmodules(1)*.

zstyle See the section ‘The zsh/zutil Module’ in *zshmodules(1)*.

ztcp See the section ‘The zsh/net/tcp Module’ in *zshmodules(1)*.

B.1. Characteristics of Linux RAID levels

All RAID levels are used to combine multiple devices into a single MD array. The MD plug-in is a region-manager, so EVMS refers to MD arrays as "regions." MD can create these regions using disks, segments or other regions. This means that it's possible to create RAID regions using other RAID regions, and thus combine multiple RAID levels within a single volume stack.

The following subsections describe the characteristics of each Linux RAID level. Within EVMS, these levels can be thought of as sub-modules of the MD plug-in.

B.1.1. Linear mode

Linear-RAID regions combine objects by appending them to each other. Writing (or reading) linearly to the MD region starts by writing to the first child object. When that object is full, writes continue on the second child object, and so on until the final child object is full. Child objects of a Linear-RAID region do not have to be the same size.

Advantage:

- Linear-RAID provides a simple method for building very large regions using several small objects.

Disadvantages:

- Linear-RAID is not "true" RAID, in the sense that there is no data redundancy. If one disk crashes, the RAID region will be unavailable, and will result in a loss of some or all data on that region.
- Linear-RAID provides little or no performance benefit. The objects are combined in a simple, linear fashion that doesn't allow for much (if any) I/O in parallel to multiple child objects. The performance of a Linear-RAID will generally be equivalent to the performance of a single disk.

B.1.2. RAID-0

RAID-0 is usually referred to as "striping." This means that data in a RAID-0 region is evenly distributed and interleaved on all the child objects. For example, when writing 16 KB of data to a RAID-0 region with three child objects and a chunk-size of 4 KB, the data would be written as follows:

- 4 KB to object 0
- 4 KB to object 1
- 4 KB to object 2
- 4 KB to object 0

Advantages:

- Like Linear-RAID, RAID-0 provides a simple method for building very large regions using several small objects.

- In general, RAID-0 provides I/O performance improvements, because it can break large I/O requests up and submit them in parallel across several disks.

Disadvantage:

- Also like Linear-RAID, RAID-0 is not "true" RAID, in the sense that there is no data redundancy (hence the name RAID "zero"). If one disk crashes, the RAID region will be unavailable, and will likely result in a loss of all data on that region.

B.1.3. RAID-1

RAID-1 is usually referred to as "mirroring." Each child object in a RAID-1 region contains an identical copy of the data in the region. A write to a RAID-1 region results in that data being written simultaneously to all child objects. A read from a RAID-1 region can result in reading the data from any one of the child objects. Child objects of a RAID-1 region do not have to be the same size, but the size of the region will be equal to the size of the smallest child object.

Advantages:

- RAID-1 provides complete data redundancy. In a RAID-1 region made from N child objects, up to N-1 of those objects can crash and the region will still be operational, and can retrieve data from the remaining objects.
- RAID-1 can provide improved performance on I/O-reads. Because all child objects contain a full copy of the data, multiple read requests can be load-balanced among all the objects.

Disadvantages:

- RAID-1 can cause a decrease in performance on I/O-writes. Because each child object must have a full copy of the data, each write to the region must be duplicated and sent to each object. A write request cannot be completed until all duplicated writes to the child objects are complete.
- A RAID-1 region with N disks costs N times as much as a single disk, but only provides the storage space of a single disk.

B.1.4. RAID-4/5

RAID-4/5 is often referred to as "striping with parity." Like RAID-0, the data in a RAID-4/5 region is striped, or interleaved, across all the child objects. However, in RAID-4/5, parity information is also calculated and recorded for each stripe of data in order to provide redundancy in case one of the objects is lost. In the event of a disk crash, the data from that disk can be recovered based on the data on the remaining disks and the parity information.

In RAID-4 regions, a single child object is used to store the parity information for each data stripe. However, this can cause an I/O bottleneck on this one object, because the parity information must be updated for each I/O-write to the region.

In RAID-5 regions, the parity is spread evenly across all the child objects in the region, thus eliminating the parity bottleneck in RAID-4. RAID-5 provides four different algorithms for how the parity is distributed. In fact, RAID-4 is often thought of as a special case of RAID-5 with a parity algorithm that simply uses one object instead of all objects. This is the viewpoint that Linux and EVMS use. Therefore, the RAID-4/5 level is often just referred to as RAID-5, with RAID-4 simply being one of the five available parity algorithms.

Advantages and disadvantages

- Like RAID-1, RAID-4/5 provides redundancy in the event of a hardware failure. However, unlike RAID-1, RAID-4/5 can only survive the loss of a single object. This is because only one object's worth of parity is recorded. If more than one object is lost, there isn't enough parity information to recover the lost data.
- RAID-4/5 provides redundancy more cost effectively than RAID-1. A RAID-4/5 region with N disks provides N-1 times the storage space of a single disk. The redundancy comes at the cost of only a single disk in the region.
- Like RAID-0, RAID-4/5 can generally provide an I/O performance improvement, because large I/O requests can be broken up and submitted in parallel to the multiple child objects. However, on I/O-writes the performance improvement will be less than that of RAID-0, because the parity information must be calculated and rewritten each time a write request is serviced. In addition, in order to provide any performance improvement on I/O-writes, an in-memory cache must be maintained for recently accessed stripes so the parity information can be quickly recalculated. If a write request is received for a stripe of data that isn't in the cache, the data chunks for the stripe must first be read from disk in order to calculate the parity. If such cache-misses occur too often, the I/O-write performance could potentially be worse than even a Linear-RAID region.

B.1.5. Multipath

A multipath region consists of one or more objects, just like the other RAID levels. However, in multipath, the child objects actually represent multiple physical paths to the same physical disk. Such setups are often found on systems with fiber-attached storage devices or SANs.

Multipath is not actually part of the RAID standard, but was added to the Linux MD driver because it provides a convenient place to create "virtual" devices that consist of multiple underlying devices.

The previous RAID levels can all be created using a wide variety of storage devices, including generic, locally attached disks (for example, IDE and SCSI). However, Multipath can only be used if the hardware actually contains multiple physical paths to the storage device, and such hardware is usually available on high-end systems with fiber- or network-attached storage. Therefore, if you don't know whether you should be using the Multipath module, chances are you don't need to use it.

Like RAID-1 and RAID-4/5, Multipath provides redundancy against hardware failures. However, unlike these other RAID levels, Multipath protects against failures in the paths to the device, and not failures in the device itself. If one of the paths is lost (for example, a network adapter breaks or a fiber-optic cable is removed), I/O will be redirected to the remaining paths.

Like RAID-0 and RAID-4/5, Multipath can provide I/O performance improvements by load balancing I/O requests across the various paths.

[Prev](#)

The MD region manager

[Home](#)

[Up](#)

[Next](#)

Creating an MD region

What does " 2>&1 " mean?

Asked 13 years, 10 months ago Modified 6 months ago Viewed 1.6m times

To combine stderr and stdout into the stdout stream, we append this to a command:

3037

2>&1



e.g. to see the first few errors from compiling g++ main.cpp :



g++ main.cpp 2>&1 | head

What does 2>&1 mean, in detail?

bash shell unix redirect

Share Follow

edited Aug 10, 2022 at 19:30

asked May 3, 2009 at 22:57



codeforester

37.6k 16 107 132



Tristan Havelick

66.4k 20 54 64

- 62 @dbr I don't think it's just bash - I believe it's a bourne shell thing; hence sh, bash, ksh, ash, dash, etc. – [guns](#) May 3, 2009 at 23:49
- 8 This is part of the redirection paragraph describing POSIX-compliant shells, or POSIX shell for short. ksh is a POSIX shell for example. See:[pubs.opengroup.org/onlinepubs/009695399/utilities/...](https://pubs.opengroup.org/onlinepubs/009695399/utilities/) – [jim mcnamara](#) Apr 4, 2013 at 2:55
- 19 This construct also works on Windows. – [Vadzim](#) Oct 22, 2013 at 13:45
- 7 It's generally better doing 2>&1 than [2>/dev/null](#) ;-) – [F. Hauri - Give Up GitHub](#) Dec 8, 2013 at 12:11
- 15 I thought I'd mention that |& is shorthand for 2>&1 | if you're using zsh. I can't speak to whether that applies to other bourne-like shells or if it's a zsh only feature. – [chrixian](#) Dec 17, 2013 at 5:20

Sorted by:

19 Answers

Highest score (default)



File descriptor 1 is the standard output (stdout).



File descriptor 2 is the standard error (stderr).

3383

At first, 2>1 may look like a good way to redirect stderr to stdout . However, it will actually be interpreted as "redirect stderr to a file named 1 ".

What is the difference between /etc/fstab and /etc/mtab?

Asked 7 years, 10 months ago Modified 5 years, 4 months ago Viewed 98k times

Both `/etc/mtab` and `/etc/fstab` contain data about mounted volumes, for example:

45

`/etc/mtab`

```
/dev/xvda1 / ext4 rw,discard 0 0
proc /proc proc rw,noexec,nosuid,nodev 0 0
...
```

`/etc/fstab`

```
LABEL=cloudimg-rootfs / ext4 defaults,discard 0 0
/dev/xvdf /home/ubuntu/logs ext4 rw 0 0
```

What is the difference between the files?

`mount` `fstab`

Share Improve this question Follow

edited Apr 6, 2016 at 4:45

asked Apr 5, 2016 at 14:42

 mruru 197k ● 54 □ 484 ● 738

 Adam Matan 12.4k ● 24 □ 71 ● 90

2 Answers

Sorted by: Highest score (default) ▾

`/etc/fstab` is a list of filesystems to be mounted at boot time. If you want your Windows or file-storage partitions mounted once your computer boots, you'll need to put appropriate entries into `/etc/fstab`.

47

`/etc/mtab` is a list of *currently* mounted filesystems. If you have a disk connected but not mounted, it won't show up in the `/etc/mtab` file. Once you mount it, it will show up there.

Note also, that with systemd (to which Ubuntu switched beginning from 15.04 release) it is possible to declare filesystems that need to be mounted at boot via `*.mount` files. See [James O'Guya's tutorial](#) on the topic.

For more info, read [mount manual](#).

Share Improve this answer Follow

edited Oct 18, 2018 at 11:43

answered Apr 6, 2016 at 5:13

 slm 2,975 ● 1 □ 26 ● 33

 Sergiy Kolodyazhny 105k ● 20 □ 278 ● 496

Also note that on modern systems `/etc/mtab` is normally not written to disk anymore. Instead, it is a symbolic link pointing to `/proc/self/mounts`, which is a virtual file whose contents are generated by the kernel. – [Bachsau](#) Jul 7, 2021 at 10:12

TL;DR

30

- `/etc/fstab` is created by the user. It contains list of volumes to be mounted by `mount`.
- `/etc/mtab` is created by the system. It contains a list of currently mounted devices.
- The format of the files is similar. After mounting a new device, copy the relevant line from `/etc/mtab` to `/etc/fstab` so that it will be auto-mounted after boot or when calling `mount -a`.

Quotes from the `mount` manual

The `/etc/fstab`, `/etc/mtab` and `/proc/mounts` files

The file `/etc/fstab` may contain lines describing what devices are usually mounted where, using which options.

The programs `mount` and `umount` maintain a list of currently mounted filesystems in the file `/etc/mtab`.

When the proc filesystem is mounted (say at `/proc`), the files `/etc/mtab` and `/proc/mounts` have very similar contents. The former has somewhat more information, such as the mount options used, but is not necessarily up-to-date.

`mount -a`

`mount -a [-t type] [-O optlist]`

(usually given in a bootscript) causes all filesystems mentioned in fstab (of the proper type and/or having or not having the proper options) to be mounted as indicated, except for those whose line contains the `noauto` keyword. Adding the `-F` option will make `mount` fork, so that the filesystems are mounted simultaneously.

Share Improve this answer Follow

edited Jun 12, 2020 at 14:37

 Community Bot 1

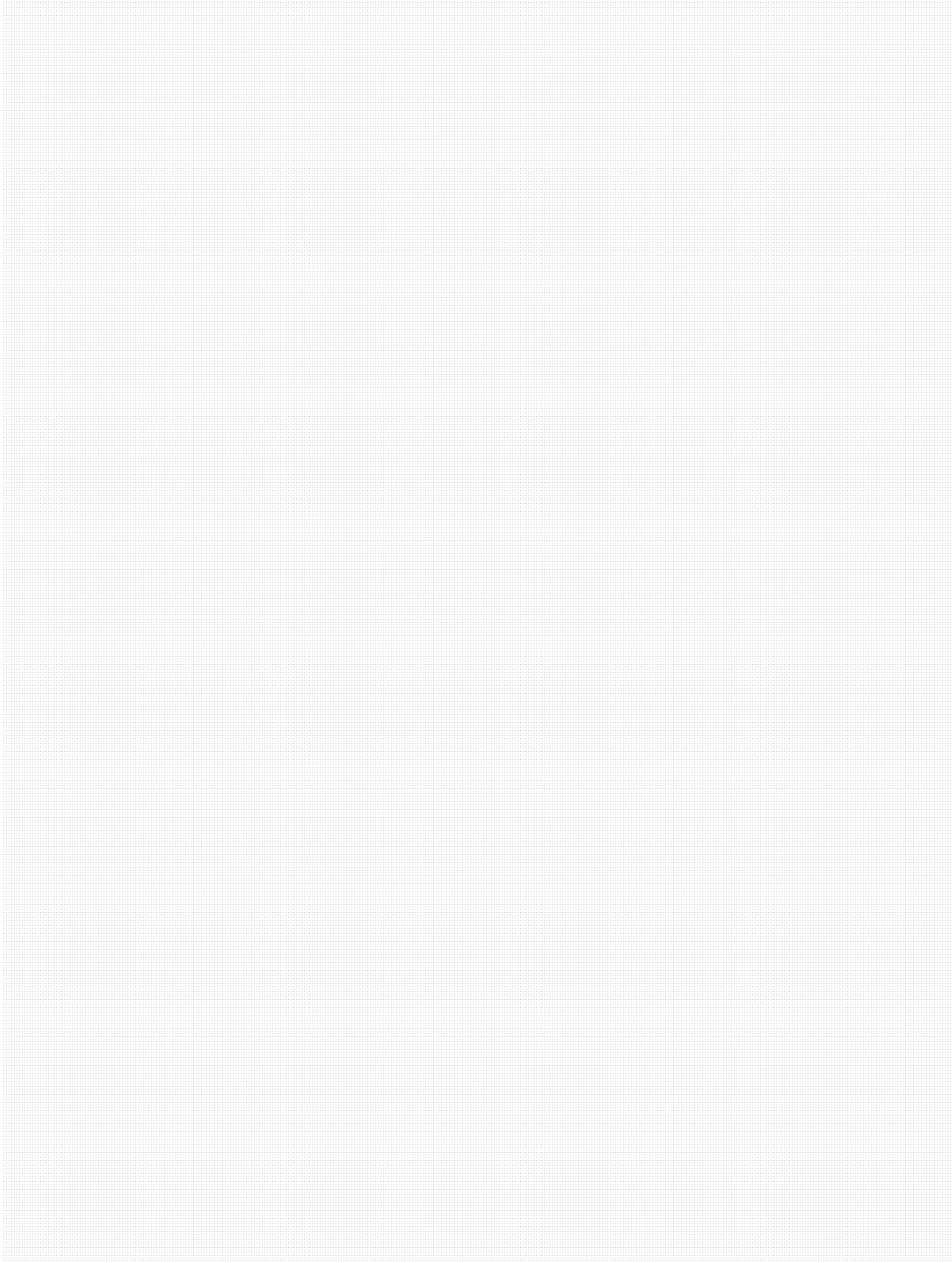
answered Apr 5, 2016 at 14:42

 Adam Matan 12.4k ● 24 □ 71 ● 90

1 What if a device shows up as mounted in `/etc/fstab` but not under `/etc/mtab`? What is going on then? – [JohnyTex](#) Nov 25, 2020 at 15:02

2 @JohnyTex it means that the user intended to mount it, but it did not happen for some reason: it might be invalid, or that `mount` was not executed. – [Adam Matan](#) Nov 25, 2020 at 19:11

1 In my particular case I can access the partition even though it is not visible in `/etc/mtab` - any idea how that is possible? – [JohnyTex](#) Nov 26, 2020 at 9:04



Mdstat

From Linux Raid Wiki

Contents

- 1 /proc/mdstat
 - 1.1 querying the status
 - 1.2 Personalities line
 - 1.3 md device line
 - 1.4 md config/status line
 - 1.5 bitmap line
 - 1.6 recovery

/proc/mdstat

The **/proc/mdstat** file shows a snapshot of the kernel's RAID/md state.

querying the status

The kernel md state is easily viewed by running:

```
cat /proc/mdstat
```

It won't hurt. Let's learn how to read the file. Here are some examples:

example 1:

```
Personalities : [raid1] [raid6] [raid5] [raid4]
md_d0 : active raid5 sde1[0] sdf1[4] sdb1[5] sdd1[2] sdc1[1]
        1250241792 blocks super 1.2 level 5, 64k chunk, algorithm 2 [5/5] [UUUUU]
        bitmap: 0/10 pages [0KB], 16384KB chunk

unused devices: <none>
```

example 2:

```
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdal[0] sdd1[2] sdb1[1]
        1465151808 blocks level 5, 64k chunk, algorithm 2 [4/3] [UUU_]
        unused devices: <none>
```

example 3:

```
Personalities : [raid1] [raid6] [raid5] [raid4]
md1 : active raid1 sdb2[1] sda2[0]
        136448 blocks [2/2] [UU]

https://raid.wiki.kernel.org/index.php/Mdstat
```

```
md2 : active raid1 sdb3[1] sda3[0]
      129596288 blocks [2/2] [UU]

md3 : active raid5 sd1l[9] sd1k[8] sd1j[7] sd1l[6] sdh1[5] sdg1[4] sdf1[3] sde1[2] sdd1[1] sdc1[0]
      1318680576 blocks level 5, 1024k chunk, algorithm 2 [10/10] [UUUUUUUUUUUU]

md0 : active raid1 sdb1[1] sda1[0]
      16787776 blocks [2/2] [UU]

unused devices: <none>
```

example 4:

```
Personalities : [raid1] [raid6] [raid5] [raid4]
md127 : active raid5 sdh1[6] sdg1[4] sdf1[3] sde1[2] sdd1[1] sdc1[0]
      1464725760 blocks level 5, 64k chunk, algorithm 2 [6/5] [UUUUU_]
      [==>.....] recovery = 12.6% (37043392/292945152) finish=127.5min speed=33440K/sec

unused devices: <none>
```

example 5:

```
Personalities : [linear] [raid0] [raid1] [raid5] [raid4] [raid6]
md0 : active raid6 sdf1[0] sde1[1] sdd1[2] sdc1[3] sdb1[4] sda1[5] hdb1[6]
      1225557760 blocks level 6, 256k chunk, algorithm 2 [7/7] [UUUUUUU]
      bitmap: 0/234 pages [0KB], 512KB chunk

unused devices: <none>
```

example 6:

```
Personalities : [raid1]
md1 : active raid1 sde1[6](F) sdg1[1] sdb1[4] sdd1[3] sdc1[2]
      488383936 blocks [6/4] [_UUUU_]

unused devices: <none>
```

Personalities line

The "Personalities" line tells you what RAID level the kernel currently supports. This can be changed by either changing the raid modules or recompiling the kernel. Possible personalities include: [raid0] [raid1] [raid4] [raid5] [raid6] [linear] [multipath] [faulty]

Note: [faulty] is a diagnostic personality; it does not mean there is a problem with the array.

md device line

Each array is then described. from example 1:

```
md_d0 : active raid5 sde1[0] sdf1[4] sdb1[5] sdd1[2] sdc1[1]
```

This means we're looking at the device /dev/md_d0.

It is active or 'started'. An inactive array is usually faulty. Stopped arrays aren't visible here.

It is a raid5 array and the component devices are:

```
/dev/sde1 is device 0
/dev/sdf1 is device 4
/dev/sdb1 is device 5
/dev/sdd1 is device 2
/dev/sdc1 is device 1
```

The order in which the devices appear in this line means nothing.

The raid role numbers [#] following each device indicate its role, or function, within the raid set. Any device with "n" or higher are spare disks. 0,1,...,n-1 are for the working array. Notice that there is no device 3. At some point that device will have failed and been replaced by device 5. (check if this is true!)

To identify a spare devices, first look for the [#/#] value on a line. The first number is the number of a complete raid device as defined. Lets say it is "n".

Also, if you have a failure, the failed device will be marked with (F) after the [#] (see example 6 where sde1 has failed. The spare that replaces this device will be the device with the lowest role number n or higher that is not marked (F). Once the resync operation is complete, the device's role numbers are swapped.

To identify the spare devices, first determine the number of device the array needs to be fully operational (see below for the [#/#] value). Lets say it is "m". The raid role numbers [#] following each device indicate its role, or function, within the raid set. Any device with "m" or higher are spare disks. 0,1,...,m-1 are for the working array.

md config/status line

The next line continues the decription of the array; in example 1 it is:

```
1250241792 blocks super 1.2 level 5, 64k chunk, algorithm 2 [5/5] [UUUUUU]
```

This line provides some basic data about the fixed size and layout: it indicates the useable size of the array in blocks is 1250241792; the array uses a 1.2 superblock and confirms a level 5 (this is redundant!) array with a chunk size of 64k using algorithm 2. (See RAID Creation for more details).

The final 2 entries on this line

```
[5/5]
[UUUUUU]
```

are more dynamic.

[n/m] means that ideally the array would have n devices however, currently, m devices are in use. Obviously when m >= n then things are good.

The

```
[UUUUUU]
```

represents the status of each device, either U for up or _ for down. So examples 2 and 6 show 'degraded' arrays with some devices 'down'.

bitmap line

If an array has a bitmap then this line describes the state. Example 1 shows

```
bitmap: 0/10 pages [0KB], 16384KB chunk
```

What would it mean when it's, eg: 23/234

This refers to the in-memory bitmap (basically a cache of what's in the on-disk bitmap – it allows bitmap operations to be more efficient).

If it's 23/234 that means there are 23 of 234 pages allocated in the in-memory bitmap. The pages are allocated on demand, and get freed when they're empty (all zeroes). The in-memory bitmap uses 16 bits for each bitmap chunk to count all ongoing writes to the chunk, so it's actually up to 16 times larger than the on-disk bitmap.

External bitmap references look like this:

```
bitmap: 5/113 pages [20KB], 8192KB chunk, file: /WIBS/<node>:md0/WIB_<node>:md0
```

recovery

example 4 is clearly showing some recovery activity:

```
[==>.....] recovery = 12.6% (37043392/292945152) finish=127.5min speed=33440K/sec
```

The first part is simply a graphical representation of the progress. The rest of the line is fairly self explanatory. The finish time is only an approximation since the resync speed will vary according to other I/O demands. See the resync page for more details.

Retrieved from "<https://raid.wiki.kernel.org/index.php?title=Mdstat&oldid=4731>"

- This page was last modified on 5 October 2013, at 09:39.

```
`-l'  
`--format=long'  
`--format=verbose'  
    In addition to the name of each file, print the file type, file  
    mode bits, number of hard links, owner name, group name, size, and  
    timestamp (*note Formatting file timestamps::), normally the  
    modification time. Print question marks for information that  
    cannot be determined.
```

Normally the size is printed as a byte count without punctuation,
but this can be overridden (*note Block size::). For example, ` -h'
prints an abbreviated, human-readable count, and
`--block-size='1'" prints a byte count with the thousands
separator of the current locale.

For each directory that is listed, preface the files with a line
`total BLOCKS', where BLOCKS is the total disk allocation for all
files in that directory. The block size currently defaults to 1024
bytes, but this can be overridden (*note Block size::). The
BLOCKS computed counts each hard link separately; this is arguably
a deficiency.

The file type is one of the following characters:

```
`-'  
    regular file  
  
'b'  
    block special file  
  
'c'  
    character special file  
  
'C'  
    high performance ("contiguous data") file  
  
'd'  
    directory  
  
'D'  
    door (Solaris 2.5 and up)  
  
'l'  
    symbolic link  
  
'M'  
    off-line ("migrated") file (Cray DMF)  
  
'n'  
    network special file (HP-UX)
```

```
`p'  
    FIFO (named pipe)  
  
'P'  
    port (Solaris 10 and up)  
  
's'  
    socket  
  
'?'  
    some other file type
```

The file mode bits listed are similar to symbolic mode specifications (*note Symbolic Modes::). But `ls' combines multiple bits into the third character of each set of permissions as follows:

```
's'  
    If the set-user-ID or set-group-ID bit and the corresponding executable bit are both set.  
  
'S'  
    If the set-user-ID or set-group-ID bit is set but the corresponding executable bit is not set.  
  
't'  
    If the restricted deletion flag or sticky bit, and the other-executable bit, are both set. The restricted deletion flag is another name for the sticky bit. *Note Mode Structure::.  
  
'T'  
    If the restricted deletion flag or sticky bit is set but the other-executable bit is not set.  
  
'x'  
    If the executable bit is set and none of the above apply.  
  
'-'  
    Otherwise.
```

Following the file mode bits is a single character that specifies whether an alternate access method such as an access control list applies to the file. When the character following the file mode bits is a space, there is no alternate access method. When it is a printing character, then there is such a method.

GNU `ls' uses a `.' character to indicate a file with an SELinux security context, but no other alternate access method.

A file with any other combination of alternate access methods is marked with a `+' character.

On a regular (MBR) system, there are a maximum of 4 reg. partitions.
On a GPT system, you can have up to 128 partitions.